НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

_____ Сергій СТІРЕНКО

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Система безпеки розумного дому з використанням web-socket (клієнтська частина)»

Виконав:
студент IV курсу, групи ІО-63
Алегрі Владислав Віталійович _____

Керівник:
асистент кафедри ОТ
Стешин Віктор Васильович _____

Консультант з нормоконтролю:
Професор кафедри ОТ, д.т.н.
Сімоненко Валерій Павлович _____

Рецензент:

_____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій СТІРЕНКО

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Алегрі Владиславу Віталійовичу

1. Тема проєкту «Система безпеки розумного дому з використанням web-socket», керівник проєкту Стешин Віктор Васильович, асистент кафедри ОТ, затверджені наказом по університету від «07» травня 2020 р. № 1081-с

2. Термін подання студентом проєкту 26 травня 2020р.

3. Вихідні дані до проєкту див. технічне завдання

4. Зміст пояснювальної записки дослідження предметної області, огляд існуючих рішень, визначення вимог і завдань для програмного продукту, вибір платформи та технології, обґрунтування оптимальності використання обраних інструментів для розробки, реалізація проєкту.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) схема функціональна, схема принципова, схема структурна

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Сімоненко В.П.		

* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломного проєкту.

7. Дата видачі завдання 01.09.2019

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Відмітки про виконання
1	*Затвердження теми роботи*	01.09.2019	виконано
2	*Вивчення та аналіз завдання*	02.09.2019-02.02.2020	виконано
3	*Розробка архітектури та загальної структури систем*	03.02.2020-03.03.2020	виконано
4	*Розробка структур окремих Підсистем*	04.03.2020-15.03.2020	виконано
5	*Програмна реалізація системи*	16.03.2020-12.04.2020	виконано
6	*Оформлення пояснювальної записки*	13.04.2020-17.05.2020	виконано
7	*Захист програмного продукту*	25.04.2020	виконано
8	*Передзахист*	26.05.2020	виконано
9	*Захист*	15.06.2020	

Студент Владислав АЛЕГРІ

Керівник Віктор СТЕШИН

Анотація

У бакалаврській дипломній роботі реалізовано систему безпеки розумного дому з використанням web-socket, призначеної для забезпечення безпеки приміщення користувача.

Програма дозволяє керувати пристроями розумного дому, а також отримувати інформацію про їх стан. Програмний продукт був реалізований на мові Javascript за допомогою бібліотеки ReactJS у візуальному середовищі Visual Studio Code.

Аннотация

В данной бакалаврской дипломной работе реализована система безопасности умного дома с использованием web-socket, предназначенной для обеспечения безопасности помещения пользователя.

Программа позволяет управлять устройствами умного дома, а также получать информацию о их состоянии. Программный продукт был реализован на языке Javascript с помощью библиотеки ReactJS в визуальной среде Visual Studio Code.

Annotation

In this work for a Bachelor's Degree, the smart home security system using a web-socket is realized to ensure the safety of the user's premises.

The software product makes it possible to manage smart home devices, as well as receive information about their status. The software product was realized in the Javascript language using the ReactJS library in the Visual Studio Code visual environment.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ІАЛЦ.467100.002 ТЗ	Система безпеки розумного дому з використанням web-socket (клієнтська частина). Технічне завдання	4	
3	А4	ІАЛЦ.467100.003 ПЗ	Система безпеки розумного дому з використанням web-socket (клієнтська частина). Пояснювальна записка	58	
4	А4	ІАЛЦ.467100.004 А1	Система безпеки розумного дому з використанням web-socket (клієнтська частина). Схема структурна – структура програми	1	
5	А4	ІАЛЦ.467100.005 А2	Система безпеки розумного дому з використанням web-socket (клієнтська частина). Схема функціональна – схема прецедентів	1	
6	А4	ІАЛЦ.467100.006 А3	Система безпеки розумного дому з використанням web-socket (клієнтська частина). Схема принципова – схема алгоритму додавання нового пристрою	1	

Технічне завдання
до дипломного проєкту

на тему: «Система безпеки розумного дому з використанням web-socket (клієнтська частина)»

Київ – 2020

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання розповсюджується на розробку клієнтської частини системи безпеки розумного дому з використанням web-socket.

Область застосування: альтернатива сучасним системам безпеки розумного дому.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки служить завдання на виконання розробки клієнтської частини системи безпеки розумного дому з використанням web-socket, затвердженою кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний Інститут ім. Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проєкту є розробка клієнтської частини системи безпеки розумного дому з використанням web-socket.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами для розробки служать науково-технічна література з комп’ютерних технологій, публікації в періодичних виданнях, публікації в Інтернеті за даним питанням.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

- Розробка інтерфейсу для входу та реєстрації користувачів;
- Розробка інтерфейсу для керування пристроями;
- Розробка інтерфейсу для додавання нових пристроїв.

5.2. Вимоги до програмного програмного забезпечення

- Операційна система Windows, Linux, macOS
- Остання версія браузерів Chrome, Mozilla

6. ЕТАПИ РОЗРОБКИ

	Дата
Затвердження теми роботи	01.09.2020
Вивчення та аналіз завдання	02.09.2019-02.02.2020
Розробка архітектури та загальної структури систем	03.02.2020-03.03.2020
Розробка структур окремих підсистем	04.03.2020-15.03.2020
Програмна реалізація системи	16.03.2020-12.04.2020
Оформлення пояснювальної записки	13.04.2020-17.05.2020
Захист програмного продукту	
Передзахист	26.05.2020
Захист	

Пояснювальна записка
до дипломного проєкту

на тему: «Система безпеки розумного дому з використанням web-socket (клієнтська частина)»

Київ – 2020

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ ТА СКОРОЧЕНЬ

ОС	Операційна система
Фреймворк	Інфраструктура програмних рішень, що полегшує розробку складних систем.
JS	Мова програмування JavaScript
DOM	Об'єктна модель документа (англ. Document Object Model, DOM) — специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами
HTML	Мова розмітки гіпертекстових документів (англ. HyperText Markup Language) — стандартна мова розмітки веб-сторінок в Інтернеті.
API	Прикладний програмний інтерфейс(англ. Application Programming Interface, API)
JSON	Текстовий формат обміну даними між комп'ютерами, який дозволяє описувати об'єкти та інші структури даних(англ. JavaScript Object Notation)
SPA	Односторінковий додаток(англ. Single Page Application, SPA)
CSS	Каскадні таблиці стилів (англ. Cascading Style Sheets, CSS) — формальна мова опису зовнішнього вигляду документу
IoT	Інтернет-речей(англ. Internet of Things, IoT) — концепція обчислювальної мережі фізичних предметів («речей»), оснащених вбудованими технологіями для взаємодії іншого з іншим або з зовнішньою середою[17]

ВСТУП

Сьогодні набирають популярність системи безпеки розумного дому. Такі системи забезпечують безпеку дому та захищають від надзвичайних ситуацій. До їх складу входить охоронно-пожежна сигналізація, відеоспостереження всередині дому, відеодомофон, охорона периметру і т.п.

Актуальність теми

Через зростання популярності використання систем безпеки розумного дому, виникає потреба у розробці програм, що дозволять простіше та швидше налаштувати таку систему.

Мета і задачі дослідження

Метою роботи є розробка системи безпеки розумного дому з використанням web-socket , що дозволить простіше та швидше налаштовувати таку систему.

Для досягнення поставленої мети були поставлені наступні основні задачі:

- Провести аналіз існуючих систем безпеки розумного дому;
- Створити програмну реалізацію розробленої системи;
- Провести тестування розробленої системи.

Практичне значення

Запропонована система безпеки розумного дому полягає у простішому та швидшому часі налаштування такої системи за рахунок використання web-socket, яка дозволяє встановлювати з’єднання між клієнтом та сервером для обміну повідомленнями у режимі реального часу. Це дозволить швидше налаштовувати таку систему.

РОЗДІЛ 1.
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1. Загальні відомості

"Розумний дім" - це така система управління комфортом і безпекою будинку та його мешканців, яка призначена забезпечити безпеку будинку та захистити його від будь-яких надзвичайних ситуацій [9]. Сюди входять: захист від вторгнення за допомогою камер відеоспостереження, автоматизація дверей, воріт, охоронної сигналізації, запобігання аварійним ситуаціям.

Системи безпеки забезпечують:

- Контроль цілісності периметра (двері і вікна);
- Автоматизований контроль доступу в приміщення;
- Відеоспостереження за прилеглою територією;
- Отримання зображення з будь-якої камери відеоспостереження за допомогою Інтернету;
- Автоматичне освітлення території при проникненні.

1.1. Різновиди систем розумного дому

Системи розумного дому поділяються на:

- Дротові;
- Бездротові;
- Централізовані;
- Децентралізовані;
- З відкритим протоколом;
- З закритим протоколом.

1.1.1. Дротова система

У дротовій системі всі керуючі пристрої - датчики, вимикачі, пристрої управління кліматом, різноманітні керуючі панелі зв'язуються єдиною

дротовою інформаційною шиною, якою йдуть сигнали до виконавчих пристроїв.

Переваги дротової системи:

- Надійність;
- Швидкість відгуку;
- Різноманітність інтегрованих систем;
- Довгий термін служби;
- Пожежна безпека.

1.1.2. Бездротова система

У бездротових системах, на відміну від дротових, сигнал від керуючих пристроїв до виконавчих йде радіоканалом, а не дротами. Це дозволяє скоротити використання дротів, а також час на інсталяцію системи. Ці системи також монтуються на об'єкти з готовим ремонтом з класичною проводкою. Кожен бездротовий "вимикач" є ще і радіопередавачем, який зв'язується з усіма іншими "вимикачами". Це дозволяє створювати різні сценарії використання (нічний режим, вимкнути усе і т.п.) та перепрограмовувати функціонал клавіш.

Переваги бездротової системи:

- Можна встановлювати в квартири і будинки з уже готовим ремонтом з класичною проводкою;
- Менша кількість дротів, у порівнянні з дротовою системою;
- На відміну від дротової системи, проєктування не потребується;
- Вартість.

1.1.3. Централізована система

У централізованій системі програмується лише один логічний центральний модуль. Зазвичай це вільно програмований контролер з великою кількістю виходів. У контролер записується заздалегідь спеціально створена під об'єкт програма, за допомогою якої йде управління виконавчими пристроями та інженерними системами. Це дозволяє використовувати

широкий вибір обладнання та складних сценаріїв. Централізовані системи можуть бути як дротовими, так і бездротовими.

Переваги централізованої системи:

- Можливість управління всіма інженерним системами в єдиному інтерфейсі;
- Можливість створювати складні сценарії, пов’язані з часом доби, станом мешканця, температурою, місячним циклом;
- Можливість підключення майже будь-якого обладнання.

1.1.4. Децентралізована система

У децентралізованих системах "Розумного дому" кожен виконавчий пристрій має власний мікропроцесор з енергонезалежною пам'яттю. При виході з ладу одного пристрою вся система працює справно, крім приладів підключених до цього пристрою. Прикладом децентралізованої системи є "розумний дім" побудовані на основі протоколу KNX (найпопулярнішого в Європі).

Переваги децентралізованої системи:

- Надійність. Усі пристрої не залежать один від одного і мають енергонезалежну пам'ять.
- Популярність;
- Можливість використовувати додатковий блок логіки, який буде відповідати за специфічні сценарії;

1.1.5. Система з відкритим протоколом

Протокол - це мова якою спілкуються всі пристрої в "розумному домі". Якщо взяти протокол KNX, то він є відкритим. Багато виробників виготовляють пристрої, що працюють на цій мові. Асоціація KNX перевіряє їх на сумісність і тестує. Логотип KNX EIB на пристрої гарантує підвищену якість.

Переваги системи з відкритим протоколом:

- Великий вибір виробників. Це означає, що є великий вибір пристроїв за дизайном, ціною, характеристиками;
- Оновлення та конкуренція. Виробники конкурують в одному сегменті, що змушує їх розвиватися і винаходити нові пристрої.

1.1.6. Система з закритим протоколом

Для того, щоб спростити процес програмування, зменшити витрати на виробництво, деякі виробники випускають обладнання, яке працює на власному закритому протоколі. Крім них ніхто таке обладнання не випускає.

Переваги системи з закритим протоколом:

- Наявність цікавих рішень за нижчою ціною;
- Нижча вартість, ніж у систем з відкритим протоколом;
- Більш швидке реагування на вимоги ринку.

1.2. Існуючі системи розумного дому

1.2.1. Система Ajax

Українська система Ajax автоматизації будинку повною мірою покриває декілька значних задач: забезпечення комфорту та зручності управлінням життєзабезпеченням приміщення; гарантує безпеку житла, контролюючи кордон об'єкта на предмет злому, а також електричну, пожежну, газову та інші можливі загрози для будинку.

Обладнання «Розумний дім» Ajax працює на надійно зашифрованому і захищеному двосторонньому радіозв'язку Jeweller власної розробки, має повну автономність від електромережі завдяки резервному джерелу живлення. На рис. 1.1 зображено інтерфейс web-додатку Ajax.

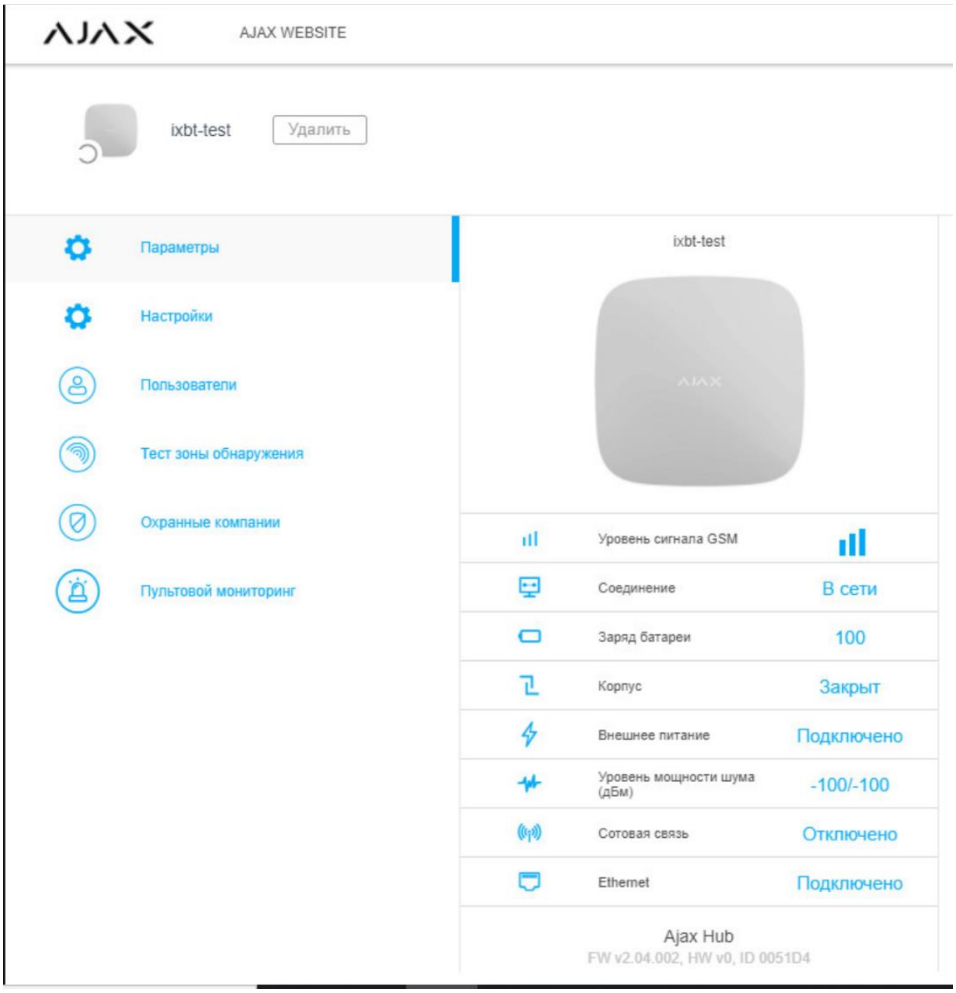


Рис. 1.1. Інтерфейс веб-додатку Ajax

Переваги системи:

- Легкий монтаж;
- Системні елементи з’єднуються бездротовим шляхом;
- Великий діапазон дії сигналу;
- Здатність працювати від вбудованого елементу живлення (до 16 годин);
- Wi-Fi та GSM-зв'язок;
- Багатий вибір способів отримання повідомлень.

Недоліки системи:

- Датчики працюють тільки з підключенням до хабу;
- Управління виключно через додаток на телефоні.

1.2.2. BroadLink система

BroadLink пропонує комплект сучасних цифрових пристроїв, які створені для керування побутовою технікою, а також охоронною системою в домі. Кожен пристрій має здатність працювати як незалежно, так і взаємодіяти з іншими елементами системи.

Переваги системи:

- Швидке налаштування;
- Не має потреби у хабі;
- Пристрої взаємодіють бездротовим шляхом;
- Керується через Wi-Fi.

Недоліки системи:

- Малий діапазон дії сигналу;
- Центральний пристрій не має резервного живлення.

1.2.3. Розумний дім Fibaro

Розумний будинок Fibaro – це професійне обладнання, яке забезпечує безпеку та автоматизацію приміщення, маючи величезну кількість опцій. Установка та налаштування системи можливе лише досвідченими фахівцями, на відміну від інших схожих систем. На рис.1.2 зображений інтерфейс додатку Fibaro.

Рис. 1.2. Інтерфейс додатку Fibaro

Переваги Fibaro:

- Користувач має багато сценаріїв використання обладнання;
- Система повідомлень;
- Взаємодіє з іншим подібним обладнанням, використовуючи протокол Z-Wave;
- Елементи системи можуть ретранслювати сигнал, збільшуючи таким чином дальність дії сигналу;
- Підтримка голосового асистента Google.

Недоліки системи:

- Установка та налаштування системи можливе лише досвідченими фахівцями;
- Для роботи потрібен центральний пристрій;
- Працює виключно через ПЗ на ПК.

1.2.4. Orvibo система

Головна задача Orvibo – забезпечення безпеки дому. Система є дешевою та простою у використанні.

Переваги системи:

- Просте підключення та установка;
- Контролюється за допомогою додатку на смартфоні;
- Автоматичне знаходження та підключення сенсорів до центрального хабу;
- Підтримка решти виробників, що дозволяє побудувати масштабну систему;
- Протокол ZigBee;
- Вартість;
- Велика кількість сценаріїв роботи.

Недоліки системи:

- Малий діапазон дії сигналу;
- Не має резервного живлення;
- Тільки дротове підключення до Інтернету.

1.2.5. Система розумного дому Xiaomi

Розумний дім від Xiaomi – доступне обладнання, що робить зручним використання всіх розумних пристроїв дому. Елементи будинку можуть працювати як незалежно, так і в системі з іншими пристроями. На рис. 1.3 зображено інтерфейс Android-додатку Xiaomi miHome.

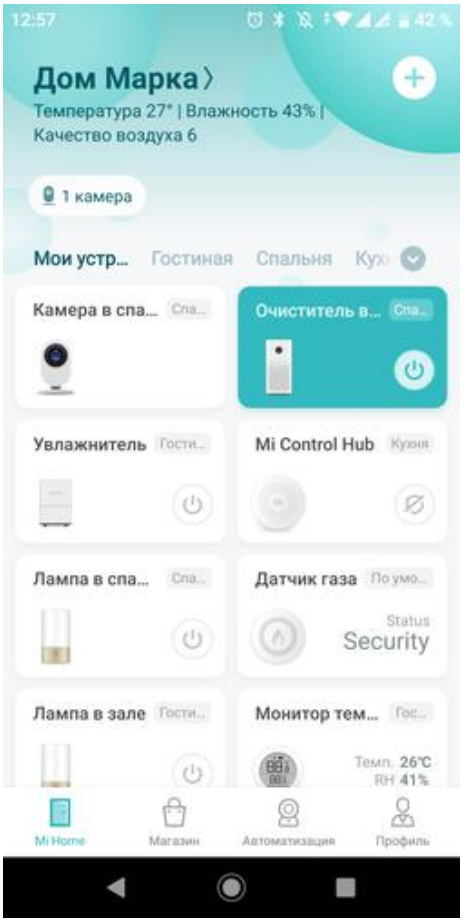


Рис. 1.3. Інтерфейс Android-додатку Xiaomi

Переваги системи:

- Незалежність пристроїв;
- Має здатність до масштабування;
- Протокол ZigBee;
- Керування через додаток на смарфоні;
- Велика кількість опцій.

Недоліки системи:

- Малий діапазон дії сигналу;
- Хаб не має резервного живлення.

ВИСНОВОК ДО РОЗДІЛУ 1

1. У даному розділі було проаналізовано системи безпеки розумного дому.
2. Було проаналізовано різновиди систем розумного дому, їх переваги та недоліки.
3. Було проаналізовані існуючі системи розумного дому, які нараз є найпопулярнішими у світі. Усі вище згадані системи схожі за функціоналом, але кожна з них має як переваги, так і недоліки.

РОЗДІЛ 2.
ПРОЄКТУВАННЯ ДОДАТКУ

2.1. Опис предметної області

Система безпеки розумного дому за своєю сутністю є поєднанням усіх систем у приміщенні користувача. Основною задачею цієї системи є забезпечення безпеки будинку та захист його від будь-яких надзвичайних ситуацій. Сюди входять: захист від вторгнення за допомогою камер відеоспостереження, автоматизація дверей, воріт, охоронної сигналізації, запобігання аварійним ситуаціям.

Кожен web-додаток повинен складатися з двох частин: клієнтської та серверної. Для організації ефективної роботи додатку існують спеціальні архітектури, що дозволяють розробляти програми з гнучким інтерфейсом користувача та можливістю подальшого розширення функціоналу.

2.1.1. Технологія Web-socket

Веб-сокети (Web-sockets) - це така технологія, за допомогою якої створюється інтерактивне з'єднання між клієнтом та сервером для обміну повідомленнями в режимі реального часу[8]. Веб-сокети, на відміну від HTTP, працюють з двонаправленим потоком даних, що робить цю технологію абсолютно унікальною.

Принцип роботи HTTP показано на рис. 2.1.

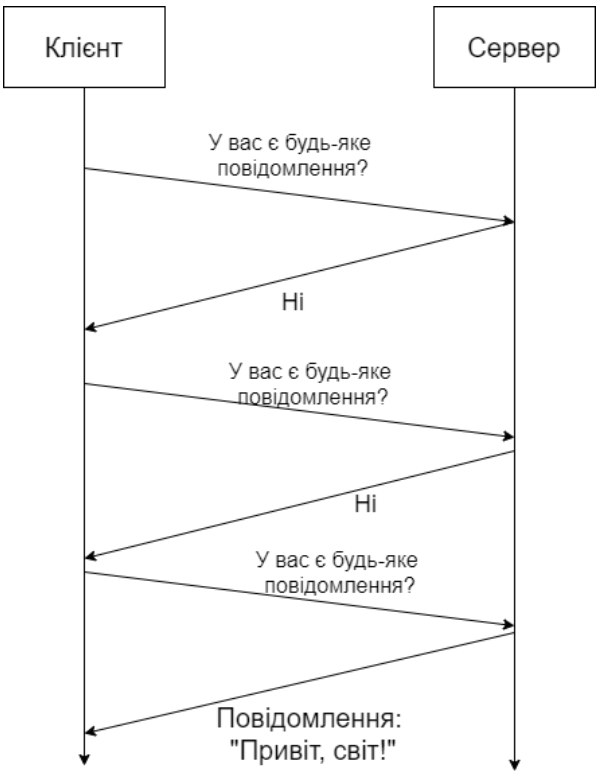


Рис. 2.1. Схема обміну повідомленнями по HTTP

2.1.2. Принцип роботи Web-socket

Веб-сокетам для відповіді не потрібні повторювані запити. Досить виконати один запит і чекати відгуку. Схема обміну повідомленнями при використанні Web-socket показана на рис. 2.2.

Веб-сокети можна використовувати, якщо ви розробляєте:

- Додатки реального часу;
- Чат-додатки;
- IoT-додатки;
- Багатокористувацькі ігри.

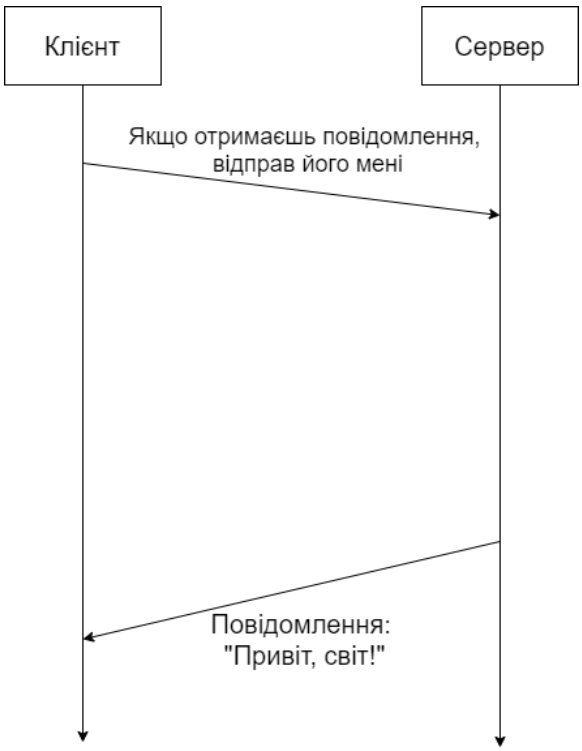


Рис. 2.2. Схема обміну повідомленнями при використанні Web-socket

2.1.3. ReactJS

React - JavaScript бібліотека, розроблена Facebook для спрощення управління інтерфейсами у веб-додатках. Особливість React у тому, що з його допомогою можна створювати керовані UI компоненти, які допомагають легко масштабувати проєкти до великих веб-додатків[7]. В основі React лежить концепція Virtual DOM - дзеркальне відображення реального DOM. Всякий раз, як вносяться зміни, React запускає процес їх вилову і оновлює справжній DOM. Концепція Virtual DOM дозволяє прискорити рендеринг додатків і підвищує продуктивність.

Головне завдання React - забезпечити виведення на екран того, що можна бачити на веб-сторінках. React значно полегшує створення інтерфейсів завдяки розподіленню кожної сторінки на невеликі фрагменти. Такі фрагменти називаються компонентами.

Компонент React - це JavaScript-функція, яка повертає шматок коду, що представляє фрагмент сторінки. Для формування сторінки функції викликаються у певному порядку, збираються разом результати викликів і показуються користувачеві.

2.1.4. VirtualDOM

Віртуальний DOM (VDOM) - це концепція програмування, в якій ідеальне або «віртуальне» уявлення призначеного для користувача інтерфейсу зберігається в пам'яті і синхронізується з «справжнім» DOM використовуючи бібліотеку ReactDOM [18]. Такий процес називається узгодженням.

Такий підхід робить API React декларативним: розробник вказує стан, у якому повинен перебувати користувацький інтерфейс, а React домагається, щоб DOM відповідав цьому стану. Це абстрагує маніпуляції з атрибутами, обробку подій і ручне оновлення DOM, які в іншому випадку довелося б використовувати при розробці програми.

Оскільки «віртуальний DOM» - це шаблон, а не конкретна технологія, цим терміном інколи позначають різні поняття. У світі React «віртуальний DOM» зазвичай асоціюється з React-елементами, оскільки вони є об'єктами, що представляють призначений для користувача інтерфейс. Проте, React також використовує внутрішні об'єкти, звані «волокнами» (fibers), щоб зберігати додаткову інформацію про дерево компонентів. Їх також можна вважати частиною реалізації «віртуального DOM» в React.

2.1.5. Порівняння React з Angular та Vue.js

Фреймворки JavaScript розвиваються дуже швидко, тому сьогодні у нас є часто оновлювані версії Angular і ReactJS, а також версії нового гравця на цьому ринку - Vue.js[10].

2.1.6. Angular 5

Angular - це фреймворк MVVM JavaScript, заснований в 2009 році, який підходить для створення інтерактивних веб-додатків.

Переваги Angular 5:

- Покращений RXJS, швидша компіляція;
- Детальна документація;
- Мінімізація можливих помилок, завдяки двосторонній прив’язці даних;

- MVVM (Model-View-ViewModel), яка дозволяє розробникам окремо працювати в одному розділі програми з використанням одного і того ж набору даних;
- Впровадження залежностей функцій пов'язаних з компонентами з модулями і модульності в цілому;

Недоліки Angular 5:

- Складний синтаксис;
- Перехід від версії до версії може викликати проблеми.

2.1.7. ReactJS

ReactJS - це бібліотека JavaScript, розроблена Facebook [11]. Такий фреймворк використовується у веб-додатках коли дані змінюються регулярно.

Переваги ReactJS:

- Легко вивчити. Завдяки простому синтаксису React дуже легко вивчити. Для написання коду потрібні лише навички HTML, які знає майже кожен веб-розробник. Не потрібно глибоко вчити Typescript, на відміну від Angular;
- Дуже гнучкий і максимально чуйний;
- Віртуальна DOM (document object model), яка дозволяє впорядковувати документи форматів HTML, XHTML або XML в дерево, яке найкраще підходить веб-браузерам задля аналізу різних елементів веб-додатку;
- Легко працює при великих навантаженнях завдяки ES 6 / 7;
- Прив’язка даних у порядку спадання. Це спеціальний потік даних, у якому дочірні елементи не можуть впливати на батьківські дані;
- JavaScript-бібліотека з відкритим сирцевим кодом, яка постійно оновлюється і покращується згідно з відгуками від розробників з усього світу;

- Невелика вага. Це досягається завдяки тому, що дані, котрі виконуються стороною користувача, мають можливість бути представленими на стороні сервера у той самий час;
- Легка система оновлень. Інженери Facebook приділяють велику увагу автоматичному оновленню і роблять цей процес максимально швидким.

Недоліки ReactJS:

- Замало якісної документації. ReactJS розроблюється швидкими темпами і у розробників залишається замало часу на написання правильної документації свого продукту. Документація наразі є невпорядкованою, адже велика кількість розробників вносять в неї зміни як їм заманеться;
- У React не поставлено чіткої мети. Через це розробники іноді мають широкий вибір тих чи інших інструментів, що викликає у них непорозуміння.
- Потрібно багато часу на освоєння. Не зважаючи на те, що синтаксис React достатньо простий, потрібне гарне розуміння того, як інтегрувати користувацький інтерфейс в структуру MVC.

2.1.8. Vue.js

Vue.js - це фреймворк JavaScript, запущений в 2013 році, який підходить для створення адаптивних користувальницьких інтерфейсів і складних односторінкових додатків.

Переваги Vue.js:

- Інтенсивний HTML. Характеристики Vue.js дуже схожі на Angular. Завдяки цьому можна оптимізувати обробку HTML-блоків з використанням різних компонентів;
- Вичерпна документація. Vue.js має дуже грунтовну документацію, яка пришвидшує навчання розробників і заощаджує багато часу на розробку програми з використанням базових знань HTML і JavaScript;

- Адаптивність. Завдяки схожості у дизайні та архітектурі Vue.js з іншими фреймворками, такими як Angular і React, швидкість переходу від інших фреймворків до Vue.js зростає;
- Файна інтеграція. Vue.js може бути використаний для створення SPA-додатка, так і для більш складних веб-додатків. Невеликі інтерактивні елементи можуть бути легко інтегровані в інфраструктуру, не наражаючи негативний вплив на всю систему;
- Широке масштабування. Vue.js допомагає розробляти багаторазові моделі, які можна розробляти, не витрачаючи багато часу завдяки простій структурі;
- Малий розмір. Vue.js може важити приблизно 15 КБ, зберігаючи при цьому свою швидкість і гнучкість. Це дозволяє досягти набагато більш високої продуктивності в порівнянні з іншими платформами.

Недоліки Vue.js:

- Нестача засобів. Vue.js, як і раніше, має досить невелику частку ринку в порівнянні з React або Angular. Це означає, що обмін знаннями в цій структурі все ще формується;
- Ризик надмірної гнучкості. Іноді у Vue.js. можуть виникнути проблеми при інтеграції в більші проєкти, так як поки немає досвіду можливих рішен;
- Відсутність повної документації англійською мовою. Це призводить до деяких труднощів на різних етапах розробки. Але наразі все більше і більше матеріалів перекладаються англійською.

2.1.9. JSX

JSX або JavaScript XML - це розширення до синтаксису мови JavaScript. За зовнішнім виглядом схожий на HTML, JSX надає спосіб структурування рендеринга компонентів з використанням синтаксису, знайомого багатьом розробникам[12]. Компоненти React зазвичай пишуться з використанням JSX, хоча це не обов'язково (компоненти також можуть бути написані на чистому

JavaScript). JSX схожий на інший синтаксис розширення, створений Facebook для PHP під назвою XHP.

Переваги JSX:

- Він швидше за JavaScript, тому що виконує оптимізацію при компіляції коду в JavaScript;
- Він також пропонує безпеку типів, більшість помилок можуть бути виявлені під час компіляції;
- JSX спрощує і прискорює запис шаблонів, якщо ви знайомі з HTML.

2.1.10. Web-socket JWT

Веб-токен JSON, або JWT, являє собою стандартизований, в деяких випадках підписаний і / або зашифрований формат упаковки даних, який використовується для безпечної передачі інформації між двома сторонами[13].

JWT визначає особливу структуру інформації, яка відправляється по мережі. Вона представлена в двох формах - серіалізованій і десеріалізованій. Перша використовується безпосередньо для передачі даних із запитами і відповідями. З іншого боку, щоб читати і записувати інформацію в токен, потрібна його десеріалізація.

У несеріалізованному вигляді JWT складається з заголовка і корисного навантаження, які є звичайними JSON-об'єктами. Заголовок JOSE використовується для опису криптографічних функцій, які застосовуються для підпису та / або шифрування токена. Тут також можна вказати додаткові властивості, наприклад, тип вмісту.

Процес серіалізації JWT складається з кодування заголовка, корисного навантаження та підпису, якщо він є, за допомогою алгоритму base64url. Це проста варіація base64, яка використовує URL-безпечний символ «_» замість небезпечних «+» і «/». Справа в тому, що деякі інструменти обробки даних розпізнають керуючі символи в рядку, тому їх бути не повинно. На рис. 2.3 показано процес серіалізації непідписаного токену.

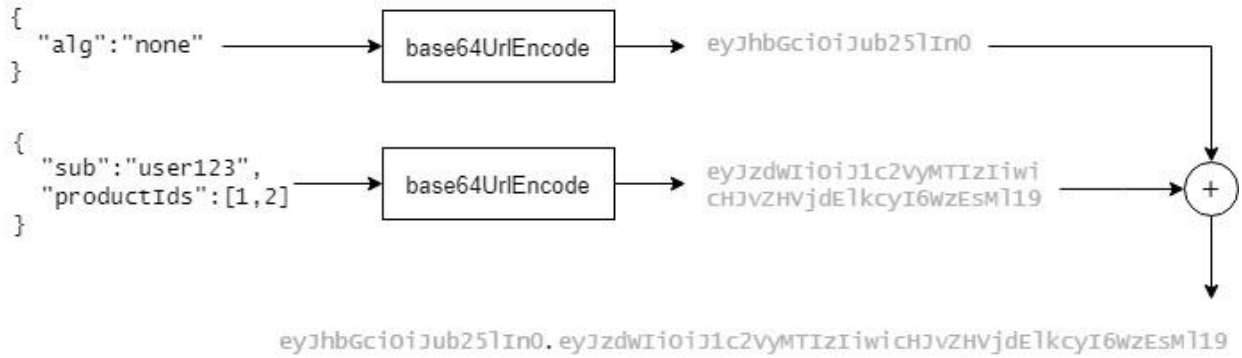


Рис. 2.3. Процес серіалізації непідписаного токену

2.1.11. Single Page Application (SPA)

Single Page Application, або «додаток однієї сторінки» - це тип web-додатку, в якому завантаження необхідного коду відбувається на одну сторінку, що дозволяє заощадити час на повторне завантаження одних і тих же елементів[14].

Обробка даних в додатку SPA відбувається на стороні сервера: призначений для користувача браузер відкриває інтерфейс програми, після чого відправляє команди програмі і отримує оброблену інформацію. Це спільна риса для всіх веб-додатків, яка, при наявності інтернет-з'єднання у користувача, дозволяє йому використовувати програмні інструменти, без скачування їх на власний пристрій.

Особливість архітектури SPA полягає в тому, що всі елементи, необхідні для роботи програмного забезпечення: елементи CSS, скрипти, стилі та ін. завантажуються на одній сторінці. Вони завантажуються при ініціалізації. Також даний вид додатків завантажує додаткові модулі після запиту від користувача. Будь-яка призначена для користувача активність фіксується для зручності навігації. Це дозволяє скопіювати посилання і відкрити софт на тому ж етапі взаємодії на іншій вкладці, браузері або пристрої.

При завантаженні нових модулів в SPA контент на них оновлюється тільки частково, так як елементам, що не потребує зміні, немає необхідності завантажуватися повторно, сповільнюючи тим самим швидкість відповіді і переданий обсяг даних між браузером і сервером.

Даний вид програмного забезпечення за способом взаємодії з користувачем найбільше схожий на роботу десктопних додатків. Але, як ми вже говорили, різниця в тому, що пристрій користувача: ПК, планшет або навіть телефон, є тільки засобом введення і виведення інформації, а самий додаток розташований на сервері і використовує його обчислювальні потужності.

2.1.12. Переваги

Web-Socket - це сучасний спосіб мати постійне з'єднання між браузером і сервером. Веб-сокети є однією з найперспективніших веб-технологій, яку вже зараз використовують багато розробників. Вона відмінно підходить для взаємодії в режимі реального часу.

- Немає обмежень, пов'язаних з крос-доменними запитами;
- Мають хорошу підтримку браузерами;
- Можуть відправляти / отримувати як рядки, так і бінарні дані;
- Простий API.

React-розробка полягає в описі того, що потрібно вивести на сторінку (а не в складанні інструкцій для браузера, присвячених тому, як це робити). Це, крім іншого, означає значне скорочення обсягів шаблонного коду.

2.2. Визначення вимог і завдань

Основними функціями системи є:

1. Додавання нових пристроїв за заданими критеріями:
   a. назва пристрою;
   b. тип пристрою;
   c. місце розташування пристрою;
2. Керування пристроями;
3. Перегляд інформації про пристрій;
4. Можливість користувачем вмикати та вимикати пристрої;
5. Забезпечення безпеки приміщень.

Основними вимогами до сервісу є:

1. Підтримка сервісу на вебсайті, адаптованому до мобільних пристроїв;
2. Зрозумілий та простий користувацький інтерфейс;
3. Додаток повинен підтримуватися не менш, ніж двома платформами;
4. Локалізація системи англійською мовою;
5. Додаток повинен бути доступним з будь-якої точки світу.

2.3. Опис функціоналу додатку

Система орієнтована для усіх людей, які користуються IoT-приладами у своїх домівках.

Функціонал для користувачів:

1. Можливість реєстрації користувача у системі. Для реєстрації потрібно ввести унікальну електронну адресу та пароль для цього облікового запису;
2. Можливість авторизації користувача. Для цього необхідно ввести свій логін та пароль, котрі були використані користувачем для реєстрації;
3. Створення, перегляд та редагування профілю;
4. Можливість керувати кожним пристроєм у системі;
5. Можливість переглядати пристрої, розподілені по приміщеннях;
6. Можливість вмикати та вимикати усі пристрої одночасно;
7. Можливість вмикати та вимикати систему безпеки у домі;
8. Можливість перегляду повідомлень;
9. Вихід зі свого профілю у системі.

2.4. Прецеденти

Згідно з вимогами та функціями, що визначені у пунктах 2.2. та 2.3., вони повинні бути реалізовані системою, проводимо деталізацію прецедентів використання веб-додатку та виконуємо побудову варіативної діаграми. На рис. 2.4 зображена діаграма прецедентів, яка демонструє основні взаємодії користувача і системи.

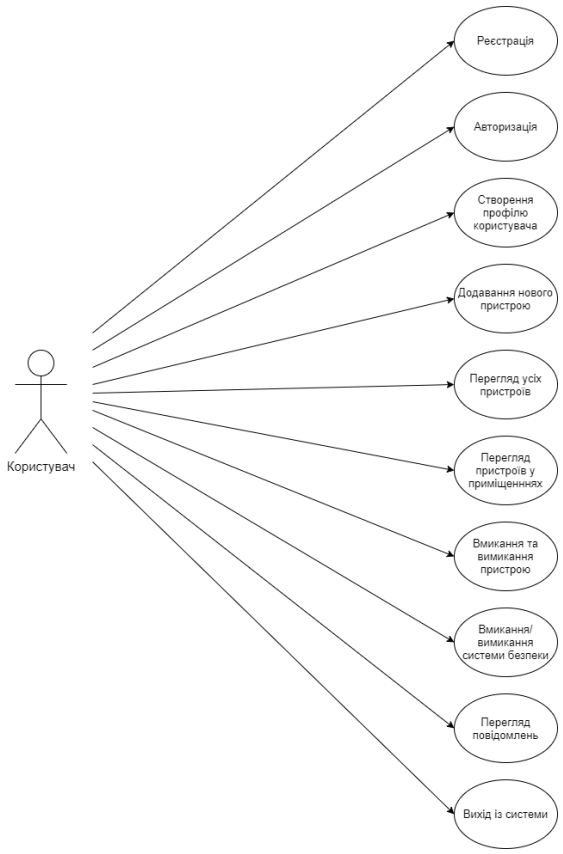


Рисунок 2.4 – Діаграма прецедентів взаємодії користувача і системи

На рисунках 2.5 – 2.10 зображені детальні ієрархії прецедентів для користувача – реєстрації, авторизації, відображення списку пристроїв у кімнаті, відображення списку всіх пристроїв, відображення списку повідомлень, додавання нового пристрою.

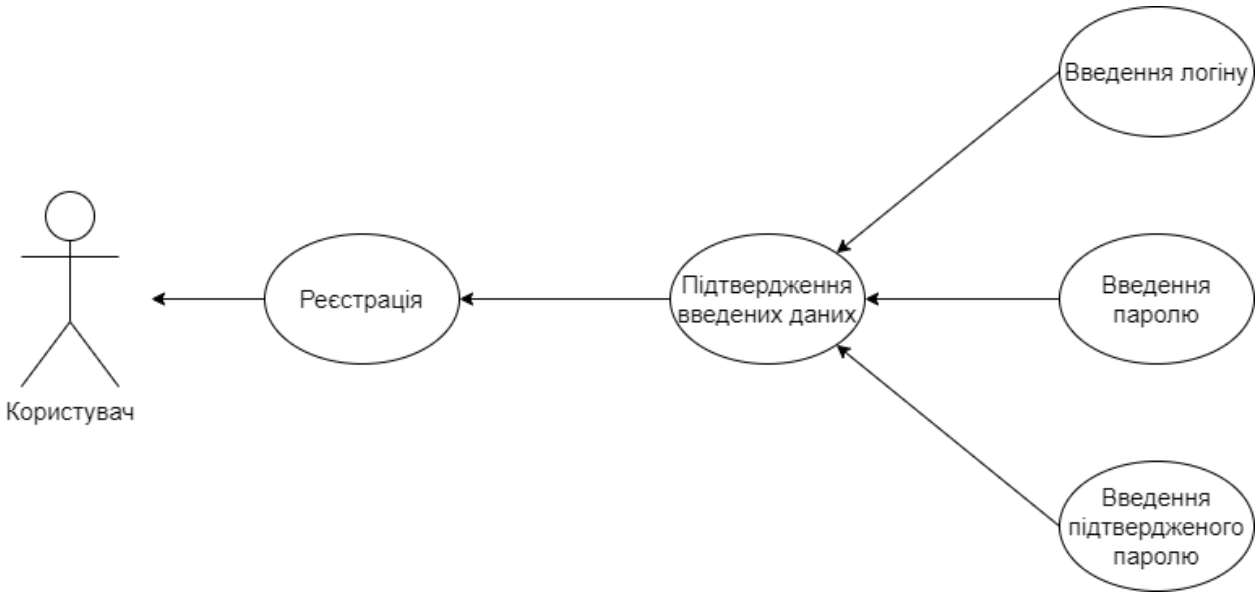


Рисунок 2.5 – Ієрархія прецеденту реєстрації користувача

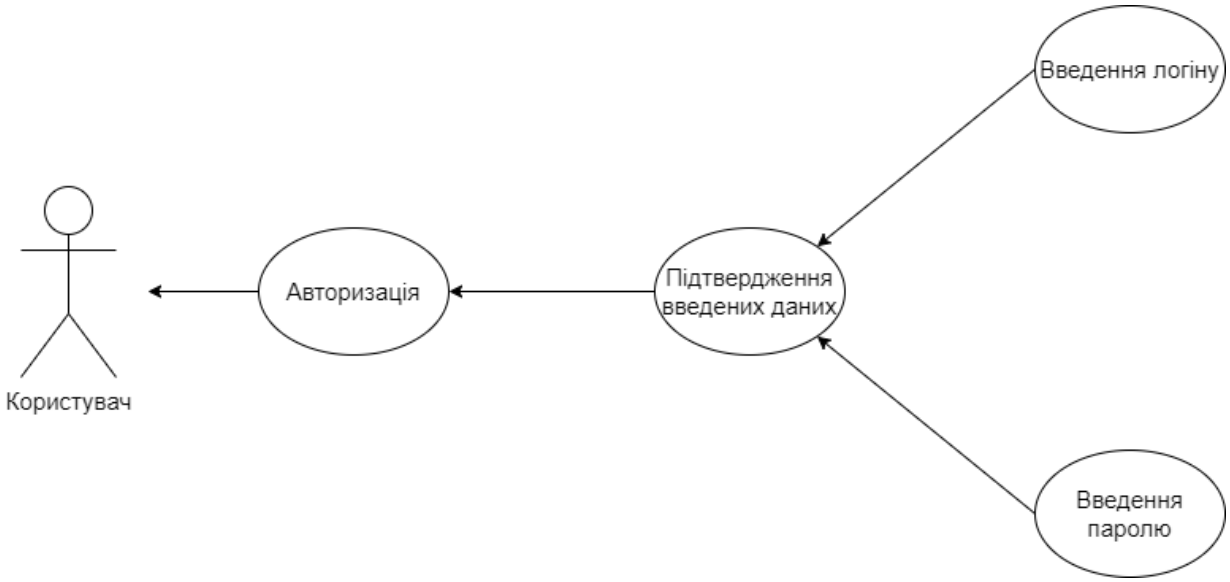


Рисунок 2.6 – Ієрархія прецеденту авторизації користувача

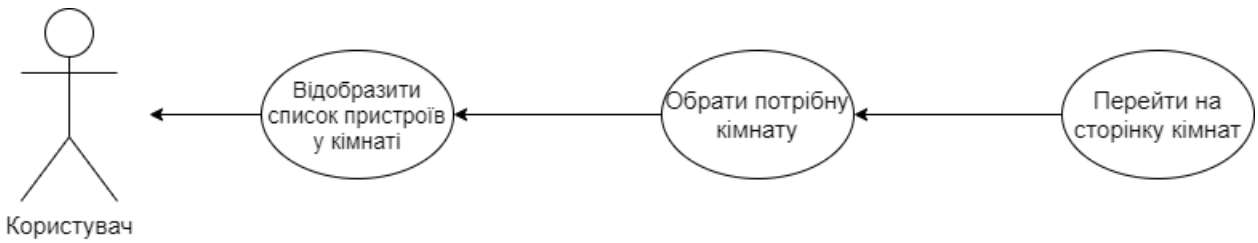


Рисунок 2.7 – Ієрархія прецеденту відображення списку пристроїв у кімнаті

Рисунок 2.8 – Ієрархія прецеденту відображення списку всіх пристроїв

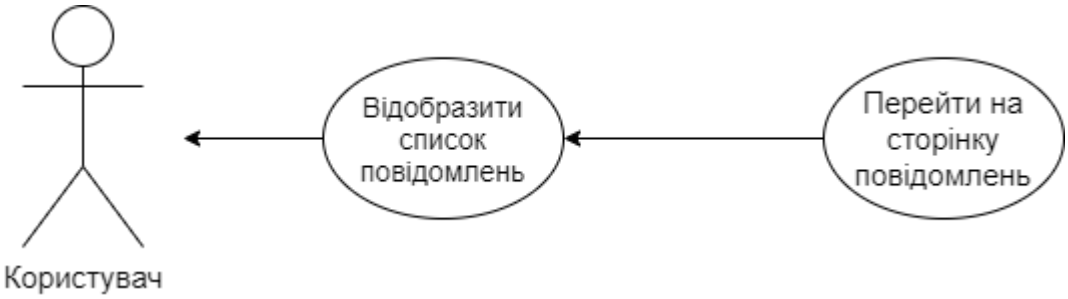


Рисунок 2.9 – Ієрархія прецеденту відображення списку повідомлень

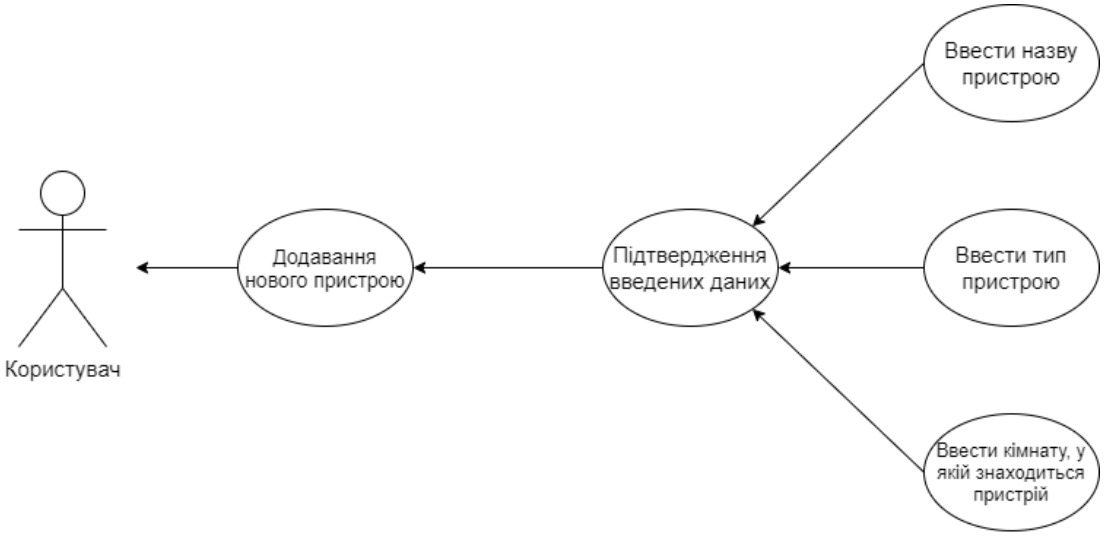


Рисунок 2.10 – Ієрархія прецеденту додавання нового пристрою

Детально проведемо розгляд сценаріїв у вищезгаданих прецедентах.

2.4.1. Прецеденти реєстрації та авторизації

Прецедент реєстрації – перша дія, котру повинен виконати новий користувач. На даному етапі потрібно ввести не зареєстровану у цій системі електронну адресу, інакше система повідомить про помилку. У табл. 2.1 представлений сценарій прецеденту реєстрації нового користувача.

Таблиця 2.1 – Реєстрація нового користувача

Назва	Реєстрація нового користувача	
Учасники	Незареєстрований користувач, система	
Передумови	Відсутні	
Результат	Реєстрація нового користувача	
Основний сценарій	Дії незареєстрованого користувача	Дії системи
1.	Введення унікальної електронно адреси (логіну) та паролю.	
2.	Натискає кнопку реєстрації.	Перевірка коректності введених даних.
3.		Додає нового користувача. Завершення процедури реєстрації.
Виключні ситуації	Некоректні дані.	

Авторизація

Даний прецедент відбувається коли користувач заходить до системи. Авторизація користувача відбувається при кожному запуску системи. Якщо користувач авторизований одразу відкривається головна сторінка додатку. У іншому випадку, користувач повертається на сторінку авторизації. У таблиці 2.2 представлений прецедент, коли користувач неавторизований у системі.

Таблиця 2.2 – Авторизація користувача

Назва	Авторизація користувача	
Учасники	Неавторизований користувач, система	
Передумови	Користувач повинен бути зареєстрованим у системі	
Результат	Авторизація успішна	
Основний сценарій	Дії неавторизованого користувача	Дії системи
1.	Введення електронної адреси(логіну) та паролю.	
2.	Натиск кнопки авторизації.	Перевіряє правильність введених даних.
3.		Перехід на головну сторінку додатку
Виключні ситуації	Некоректні дані. Такого користувача ще не зареєстровано.	

2.4.2. Прецедент відображення списку пристроїв у кімнаті

Прецедент відображення списку пристроїв у кімнаті є однією з основних дій у системі. Дія виконується, коли користувач натиснув на кнопку кімнат і обрав потрібну йому кімнату. У цій кімнаті користувач може керувати всіма пристроями приміщення. У табл. 2.3 представлений сценарій прецеденту відображення списку пристроїв у кімнаті.

Таблиця 2.3 – Список пристроїв у кімнаті

Назва	Список пристроїв у кімнаті	
Учасники	Користувач, система	
Передумови	Користувач має бути авторизованим у системі	
Результат	Відображення всіх пристроїв кімнати	
Основний сценарій	Дії користувача	Дії системи
1.	Натискає на кнопку кімнат.	Відображає список усіх кімнат.
2.	Обирає потрібну йому кімнату.	Відображає обрану користувачем кімнату.
3.		Відображає список усіх пристроїв кімнати
Виключні ситуації	Відсутні	

2.4.3. Прецедент відображення списку усіх пристроїв

Прецедент відображення списку усіх пристроїв відбувається коли користувач хоче подивитися усі IoT-пристрої його дому. За допомогою цієї дії він може керувати пристроями, не зважаючи на кімнати, у яких вони знаходяться. У табл. 2.4 представлений сценарій відображення списку усіх пристроїв.

Таблиця 2.4 – Відображення списку усіх пристроїв

Назва	Відображення списку усіх пристроїв	
Учасники	Авторизований користувач, система	
Передумови	Користувач має бути авторизованим у системі	
Результат	Отримання списку усіх пристроїв	
Основний сценарій	Дії користувача	Дії системи

Таблиця 2.4 (продовження)

Основний сценарій	Дії користувача	Дії системи
1.	Переходить на головну сторінку.	
2.	Натискає на кнопку відображення усіх пристроїв.	
3.		Відображає список усіх пристроїв.
Виключні ситуації	Відсутні	

2.4.4. Прецедент відображення повідомлень

Відображення повідомлень доступне лише авторизованим користувачам. За допомогою цієї дії користувач отримує доступ до усіх повідомлень, які надіслали йому пристрої. У табл. 2.5 представлений сценарій прецеденту відображення повідомлень.

Таблиця 2.5 – Відображення повідомлень

Назва	Відображення повідомлень	
Учасники	Користувач, система	
Передумови	Користувач повинен бути авторизованим	
Результат	Отримання усіх повідомлень пристроїв	
Основний сценарій	Дії користувача	Дії системи
1.	Переходить на головну сторінку.	
2.	Натискає кнопку повідомлень.	

Таблиця 2.5 (продовження)

Основний сценарій	Дії користувача	Дії системи
3.		Відображає усі повідомлення від пристроїв.
Виключні ситуації	Відсутні	

2.4.5. Прецедент додавання нового пристрою

Додавання нового пристрою – важлива дія для кожного користувача системи. За її допомогою користувач додає нові IoT-прилади до своєї системи. У табл. 2.6 представлений сценарій прецеденту додавання нового пристрою.

Таблиця 2.6 – Додавання нового пристрою

Назва	Додавання нового пристрою	
Учасники	Користувач, система	
Передумови	Користувач має бути авторизованим	
Результат	Успішне додавання нового пристрою	
Основний сценарій	Дії користувача	Дії системи
1.	Переходить на головну сторінку.	
2.	Натискає кнопку додавання нового пристрою.	
3.		Відображає форму для додавання нового пристрою.
4.	Введення назви, типу та кімнати, у якій буде знаходитися пристрій.	

Таблиця 2.6 (продовження)

Основний сценарій	Дії користувача	Дії системи
5.	Підтвердження вводу.	
6.		Додає новий пристрій до списку усіх пристроїв.
Виключні ситуації	Відсутні	

2.5. Розробка підходу для реалізації сервісу

Швидка взаємодія між користувачем та системою і обмін повідомленнями між ними – головна задача системи. Дані повинні оброблюватися максимально швидко, щоб користувачеві було зручно використовувати систему. Тому для розробки системи використовується бібліотека React JS та технологія web-socket.

2.5.1. Огляд ReactJS

React має невеликий API, він простий у вивченні використанні.

Елементи - це об'єкти JavaScript, які представляють HTML-елементи. Їх не існує в браузері, вони описують DOM-елементи, такі як h1, div, або section.

Компоненти - це елементи React, написані розробником. Зазвичай це частини призначеного для користувача інтерфейсу, що містять свою структуру і функціональність. Наприклад, такі як NavBar, LikeButton, або ImageUploader.

JSX - це техніка створення елементів і компонентів React. З JSX потрібно набагато менше зусиль, він трансформується в JavaScript перед запуском в браузері.

Virtual DOM - це дерево React елементів на JavaScript. React відмальовує Virtual DOM в браузері, щоб зробити інтерфейс видимим. React стежить за змінами в Virtual DOM і автоматично змінює DOM в браузері так, щоб він відповідав віртуальному.

Насамперед відбувається рендер віртуального елемента (елемента, або компонента React). Поки віртуальний елемент міститься тільки в пам'яті JavaScript, розробник повинен явно повідомити React відмалювати його в браузері.

Функція render() приймає два аргументи: віртуальний елемент і реальний вузол DOM. React бере віртуальний елемент і додає його в зазначений вузол. Після цього зображення можна побачити в браузері.

Компоненти - це душа і серце React. Це призначені для користувача елементи. Зазвичай, вони мають свою унікальну структуру і функціональність. На рис. 2.11 зображено приклад компоненту React.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Рисунок 2.11 – Приклад компоненту React

Під властивостями можна розуміти опції компонента. Вони надаються в якості аргументів компонента і виглядають так само, як атрибути HTML.

Стан - це спеціальний об'єкт всередині компонента. Він зберігає дані, які можуть змінюватись з часом.

2.6. Проєктування графічного інтерфейсу

Графічний інтерфейс системи представлений у вигляді веб-версії. Версії додатку будуть містити наступні сторінки та вікна.

2.6.1. Сторінка авторизації

Сторінка авторизації повинна містити поля для введення електронної адреси(логіну), паролю та кнопку, натиснувши яку відбудеться підтвердження авторизації. Дана сторінка з’являється у випадку, коли користувач ще не авторизований у системі. На рис. 2.12 зображено сторінку авторизації.

Рисунок 2.12 – Сторінка авторизації

2.6.2. Сторінка реєстрації

Сторінка реєстрації містить поля для вводу електронної адреси(логіну), паролю та підтвердження паролю. На рис 2.13 зображено макет сторінки реєстрації нового користувача.

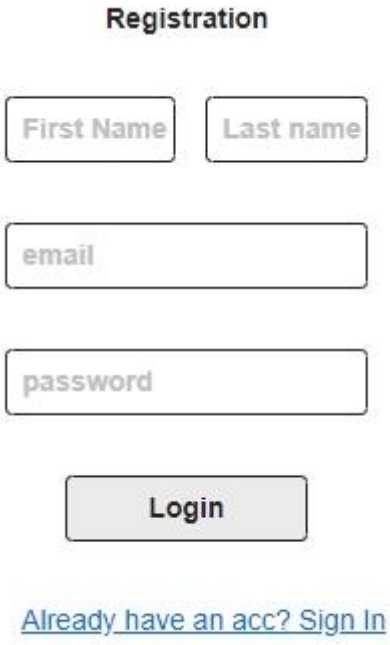


Рисунок 2.13 - Макет сторінки реєстрації

2.6.3. Головна сторінка

На головній сторінці повинні відображатися список останніх використаних приладів, кнопки управління пристроями, кнопки управління безпекою дому. Також повинно відображатися меню, яке зображене на рис. 2.15. На рис. 2.14 зображено макет головної сторінки додатку.

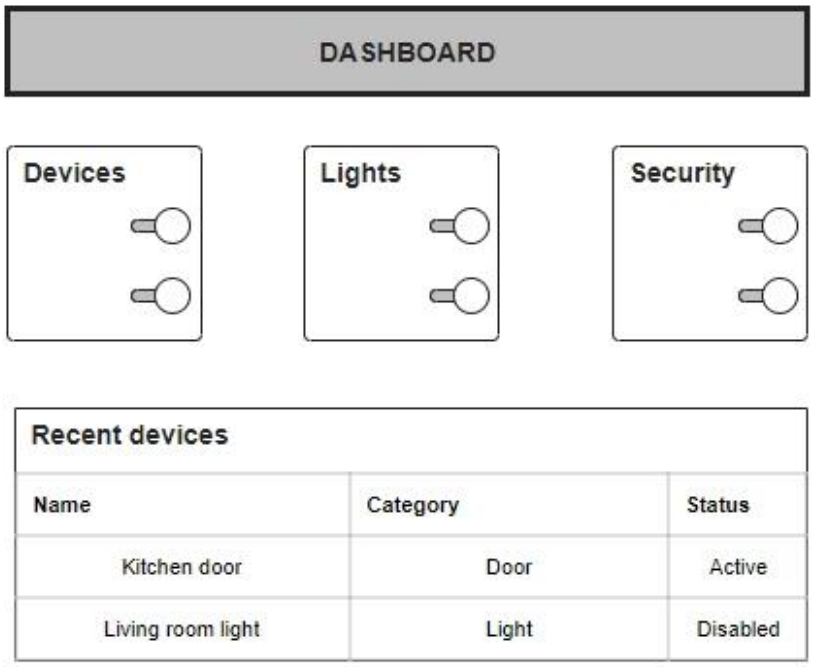


Рисунок 2.14 - Макет головної сторінки

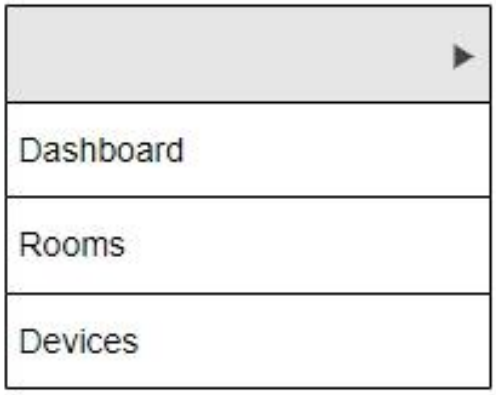


Рисунок 2.15 - Макет меню

2.6.4. Сторінка перегляду усіх приладів

Дана сторінка повинна відображати список усіх приладів розумного дому. Список повинен містити ім’я приладу, категорію, до якої цей прилад належить та статус. При натисканні кнопку користувач може вимкнути або увімкнути прилад. На рис. 2.16 зображено структуру сторінки перегляду усіх приладів.

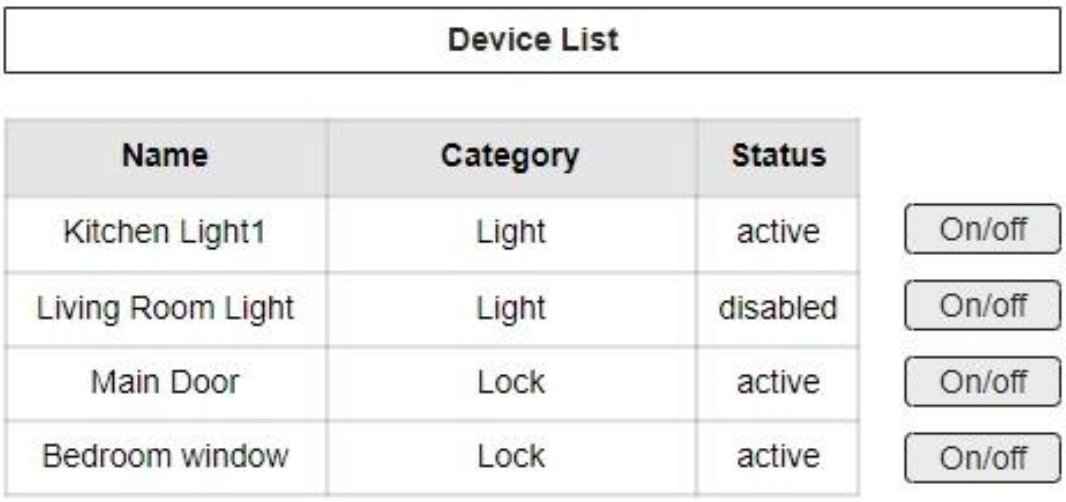


Рисунок 2.16 – Сторінка перегляду усіх приладів

2.6.5. Сторінка перегляду кімнат

Дана сторінка повинна відображати список усіх кімнат розумного дому. Сторінка повинна містити списки кімнат, які створив користувач. За допомогою цієї сторінки, користувач може переглядати статус його пристроїв у певній кімнаті, а також вмикати чи вимикати окремі пристрої. Також користувач може вимкнути усі пристрої у кімнаті, натиснувши кнопку. На рис. 2.17 зображено структуру сторінки перегляду кімнат.

Рисунок 2.17 – Сторінка перегляду усіх кімнат

ВИСНОВОК ДО РОЗДІЛУ 2

У розділі 2 було проведено опис предметної області проєкту, складені основні функції та вимоги до функціоналу додатку. Також були розроблені можливі сценарії прецедентів під час виконання основних функцій системи та визначені можливі виключні ситуації, які можуть негативно впливати на функціональність роботи системи. Це дозволить користувачу максимально комфортно використовувати систему.

Було проаналізовано три найпопулярніші фреймоворки Angular, React та Vue.js. На основі цього аналізу ReactJS було обрано для реалізації інтерфейсу системи.

На основі створених вище сценаріїв було побудовано діаграму прецедентів для користувача, а також окремі діаграмі прецедентів для детального розгляду основних функцій сервісу:

1. реєстрація;
2. авторизація;
3. додавання нового пристрою;
4. перегляд усіх пристроїв у домі;
5. перегляд усіх кімнат користувача;
6. відображення повідомлень від пристроїв.

Було розроблено графічний інтерфейс додатку веб-версії за допомогою фреймворку ReactJS та Material-UI.

РОЗДІЛ 3.
РОЗРОБКА ДОДАТКУ

3.1. Вибір технологій та їх обґрунтування

3.1.1. Вибір платформи для додатку

У розділах 1 та 2 були визначені вимоги щодо проєктування системи безпеки розумного дому з використанням web-socket та були поставлені такі завдання:

1) Додаток повинен бути легким та простим у використанні.
2) Додаток повинен підтримуватися більшістю сучасних браузерів та мобільних пристроїв.
3) Додаток повинен працювати швидко.
4) Додаток повинен приваблювати людей, які користуються IoT-пристроями.

Для того, щоб усі вимоги були дотримані, необхідно обрати платформу для розробки. Вона повинна бути зручна та швидка, щоб зробити систему якомога якіснішою. Серед запропонованих систем було виявлено такі найбільш популярні мобільні та комп'ютерні платформи:

1) Операційні системи Windows, Linux, macOS;
2) Мобільні операційні системи iOS та Android.

Розглянемо окремо кожну з перелічених систем, для того, щоб обрати таку, на якій розробка системи безпеки розумного дому буде максимально комфортною та легкою для розробника.

Android займає більше 80% ринку мобільних пристроїв[1]. Цю операційну систему встановлюють на гаджети самих різних брендів - від відомих світових лідерів до китайських «виробів». Крім того, кількість додаткових пристроїв - смарт-годинників, фітнес-браслетів і інших рішень також випускається і продається набагато більше, ніж під iOS. Це дає розробнику дуже велику аудиторію.

З іншого боку, статистика покупок через мобільні додатки така: користувачі iOS купують в 3 рази активніше. Тут аудиторія менше, але зате складається переважно з людей, які готові оплачувати якісні розробки.

Що це дає для стартапу:

1) Під Android особливо добре монетизуються безкоштовні додатки з вбудованою рекламою і окремими платними «преміум» функціями;
2) Під iOS можна сміливо створювати комерційний продукт. Але при цьому вкрай важливо переконати користувачів в його якості.

Важливо розуміти, що розробнику обов'язково мати гаджет на базі обраної операційної системи. І тут Android безумовно лідирує. Для роботи під iOS знадобиться як мінімум - iPhone, а також комп’ютер на macOS. У той час як писати під Android можна з використанням найдешевших комп'ютера і смартфона.

Існують емулятори під обидві операційні системи. Нерідко вони використовуються для розробки додатків під iOS заради економії. Але всі, хто пробував йти цим шляхом, сходяться на одному: емулятори працюють вкрай повільно і не надійно. У результаті без відповідного гаджета під рукою налагодження стає вкрай складним.

Ще один важливий фінансовий момент - додавання готового додатка в AppStore і Google Play. У першому випадку розробнику доведеться платити абонплату 99 $ на рік. Додавання нової програми або оновленої версії буде займати не один день, так як модерація в AppStore виконується вручну. Офіційний термін перевірки додатку - тиждень. А відмовити можуть іноді по абсолютно надуманим і малозрозумілим причинам. Тоді доводиться подавати додаток на модерацію повторно і знову чекати, щоб його перевірив інший модератор.

З іншого боку, конкуренція між додатками в AppStore набагато нижче, ніж в Google Play. І складна модерація - одна з причин зниження конкуренції. При цьому AppStore намагаються відсівати контент низької якості і зводить практично до нуля можливість завантаження програм з вірусами.

Google Play - безкоштовний, додавання додатки займає кілька годин. Премодерація виключно автоматична. А люди-модератори можуть звернути увагу на додаток тільки в разі великої кількості скарг.

Додатків в Google Play дуже багато, і додаються вони тисячами. Але якісних розробок набагато менше, ніж здається зі сторони. І якщо розробник зможе створити якісний продукт, то у нього є всі шанси стати популярним.

За простотою і зручністю розробки iOS лідирує відразу з кількох причин:

- Детальна і якісна документація від Apple;
- Якісний Interface Builder скорочує час роботи з інтерфейсами;
- Добре розвинене ком'юніті;
- На iOS працюють тільки Apple-пристрої, інтерфейс яких стандартизовано, проблем з відображенням додатків на різних типах пристроїв не виникає.

З мінусів називають зазвичай необхідність купувати гаджет для роботи, а також оплачувати AppStore.

Переваги розробки на Android:

- Велике співтовариство розробників стане хорошою «підмогою» новачкові;
- Гарний фреймворк, який постійно доповнюється розробниками: готові модулі знаходяться практично під будь-які потреби;
- Непогано розроблена документація і потужна підтримка від Google;
- Платформа Open source (відкритий код) дозволяє навіть без документів подивитися, що і як працює, просто заглянувши в сам код.

У числі мінусів Android-розробки:

- Середовище розробки IDE викликає масу невдоволення у розробників. Здавалося б, давно вже Android Studio коштувало налагодити. Але в Google вважають інакше. А тому система хоч і потужна, але в ній дуже багато недоробок.

- Величезна кількість пристроїв від різних брендів. Справа в тому, що кожен виробник ставить власний інтерфейс на смартфони та планшети, і вони помітно відрізняються один від одного. Крім того, з Google Play користувач може завантажити альтернативні інтерфейси. Багато в чому вони схожі, але є і важливі відмінності, через які на деяких пристроях додаток починає відображатися неправильно. Перевірити це можна тільки на практиці. Зазвичай розробники проводять тестування на максимальній кількості доступних пристроїв. І чекають перших гнівних відгуків, щоб дізнатися, на яких моделях присутні конфлікти з інтерфейсом.

Веб-додаток - клієнт-серверний додаток, в якому клієнт взаємодіє з веб-сервером за допомогою браузера [16]. Логіка веб-додатку розподілена між сервером і клієнтом, зберігання даних здійснюється, переважно, на сервері, обмін інформацією відбувається по мережі.

Переваги web-додатку:

- Користувач не потребує встановлення на свою машину великовагового програмного забезпечення. Все, що потрібно для повноцінної роботи - це браузер, який зазвичай поставляється разом з операційною системою, і доступ в Інтернет;
- Встановлюючи додатки на свій комп'ютер, мимоволі доводиться брати на себе обов'язки адміністратора, що доставляє недосвідченим користувачам масу клопоту. Додаток потрібно встановити і запустити, потім налаштувати під себе, а потім раптом можуть з’явитися випадкові помилки, які потребують негайного вирішення. У випадку з браузерним додатком, який фактично лежить на сервері, турбуватися про це не доведеться;
- Web-додатки не вимогливі до ресурсів і не мають ніяких вимог до апаратної платформи. Це означає, що немає ніякої різниці, скільки мегабайт оперативної пам'яті встановлено на комп'ютері користувача і

з якої операційної системи він працює. Потрібен лише браузер і доступ в Інтернет, все інше не так вже й важливо.

Крім того, немає ніяких проблем з підтримкою старих версій програм і зворотною сумісністю. Коли з'являється нова версія десктопного додатка, користувачам нерідко доводиться вирішувати проблеми, пов'язані з оновленням вже встановленої на їх машині копії. У випадку з браузерними додатками таких проблем не виникає - існує тільки одна версія, в якій працюють всі користувачі, і в разі виходу нової всі без винятку автоматично переходять на неї, часом навіть не помічаючи цього.

Також, web-додатки дозволяють своїм користувачам бути по-справжньому мобільними. По суті, ви можете працювати в мережі, зберігати результати своєї роботи на сервері і, в разі необхідності, мати до них доступ звідусіль.

Проаналізувавши такі платформи для розробки системи безпеки розумного дому як iOS, Android та web-додаток, було зроблено висновок, що для розробки системи доцільніше використовувати web-додаток. Такий вибір має декілька переваг:

1) Користуватися системою зможе будь-яка людина, адже веб-додаток підтримується браузером з будь-якої платформи;
2) Користувачеві не потрібно буде встановлювати додаток на свій пристрій, втрачаючи ресурси пристрою;
3) Додаток є універсальним, адже усі сучасні браузери мають мобільну версію.

3.1.2. Вибір мови програмування

Для розробки клієнтської частини системи безпеки розумного дому оберемо такі мови програмування:

1) HTML (hypertext markup language) – мова розмітки веб-сторінки, яка використовується для відображення сторінок в Інтернеті[4].
2) CSS (cascading style sheets) - мова стилів, яка описує вигляд сторінки та вказує як елементи повинні відображатися на екрані[6].

3) Javascript - це одна з найбільш популярних і потужних мов програмування, використовуваних для розробки веб-сайтів, пристосована до обміну даними з сервером, зміни стилів і вмісту веб-сторінок, керування браузерами[5].

3.1.3. Вибір допоміжних бібліотек

JavaScript – сучасна мова програмування. Для легшої та зручнішої роботи можна використовувати допоміжні фреймворки, які полегшують роботу розробнику додатку. Такими фреймворками є: Angular 5, ReactJS, Vue.js. У даній проєктній роботі буде використовуватися фреймворк ReactJS та MaterialUI.

ReactJS - це бібліотека JavaScript, розроблена Facebook. Такий фреймворк використовується у веб-додатках коли дані змінюються регулярно.

При розробці додатку використовувались такі концепції ReactJS:

1) Компоненти - найважливіша частина React. React розроблений навколо концепції багаторазових компонентів. Розробник визначає невеликі компоненти, і поєднує їх, щоб сформувати більші компоненти. Всі компоненти, маленькі чи великі, можуть використовуватися повторно, навіть у різних проєктах.
2) JSX. JSX надає спосіб структурування рендеринга компонентів з використанням синтаксису, знайомого багатьом розробникам. JSX - це компроміс, який дозволяє розробляти компоненти React використовуючи HTML-подібний синтаксис.
3) Події. React обгортає об'єкт події DOM у власний об'єкт (SyntheticEvent), щоб оптимізувати продуктивність обробки подій. Але всередині обробника подій ми все одно можемо отримати доступ до всіх методів, доступним в об'єкті події DOM. React передає обгорнутий об'єкт події для кожного виклику обробника.
4) Бібліотека React розроблена для створення користувацьких інтерфейсів і тому не включає деяких інструментів традиційного JS-фреймворка. Це дозволяє вибирати лише необхідні бібліотеки під

конкретні завдання, істотно знижуючи навантаження на додаток і сервер.

5) Бібліотека не диктує вимог до решти технологічного стека, тому ви можете створювати нові функції в React без перезапису існуючого коду. React може також функціонувати на сервері у вигляді NodeJS.

У React Material UI - частина Material Design. Material Design - це мова дизайну, вперше представлена Google в 2014 році. Це візуальна мова, яка використовує макети на основі сітки, гнучкої анімації і переходів, доповнення та ефекти глибини, такі як освітлення і тіні. Мета Material Design зводиться до трьох речей: створення, уніфікація та налаштування.

Створення - Material Design спрямований на надання візуальної мови, яка синтезує класичні принципи дизайну. Уніфікація націлена на розробку єдиної базової системи, яка об'єднує призначений для користувача інтерфейс на різних платформах, пристроях і методах введення. Налаштування забезпечує візуальну мову і гнучку основу для інновацій і брендингу.

3.2. Основні рішення з реалізації додатку та його компонентів

Розробку додатку можна поділити на такі пункти:

1. Реалізація сторінки входу;
2. Реалізація сторінки реєстрації;
3. Реалізація головної сторінки додатку, на якій можна керувати підключеними пристроями;
4. Реалізація сторінки всіх пристроїв, на якій відображається ім’я, категорія, статус пристрою та можливість додавання нових.
5. Реалізація сторінки кімнат, на якій відображаються додані користувачем кімнати, у яких знаходяться пристрої.

3.2.1. Реалізація сторінки входу

Відкриваючи сайт, користувач потрапляє на першу сторінку додатку. На цій сторінці, користувач повинен ввести електронну адресу(логін) та пароль, вказану на сторінці реєстрації. На рис. 3.1 зображена сторінка входу у додаток.

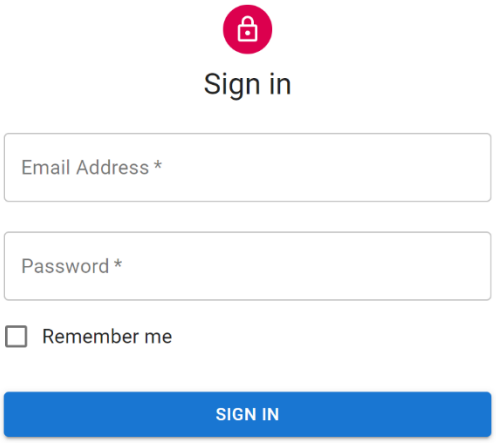


Рисунок 3.1 – Сторінка входу у додаток

3.2.2. Реалізація сторінки реєстрації у системі

Якщо користувач ще не зареєстрований у системі, користувач має можливість зареєструватися. На цій сторінці, користувач повинен ввести унікальну електронну адресу(логін) та пароль, які будуть використовуватися при вході у систему. На рис. 3.2 зображена сторінка реєстрації.

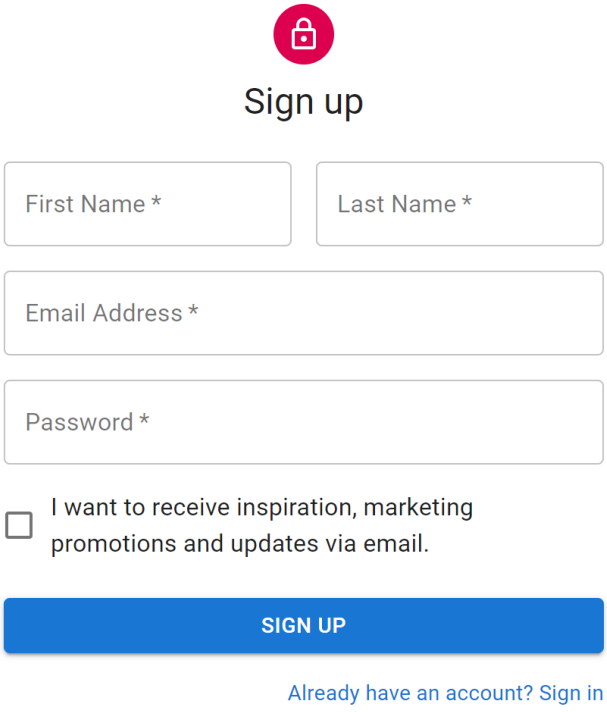


Рисунок 3.2 – Сторінка реєстрації

3.2.3. Реалізація головної сторінки додатку

Після успішного входу у систему, користувач потрапляє на головну сторінку додатку. На цій сторінці відображаються додані користувачем пристрої з декількох категорій: безпека, світло та інші пристрої. Ця сторінка ще називається панеллю приладів. На ній можна вмикати/вимикати як окремі пристрої, так і цілу категорію. На рис. 3.3 показана головна сторінка додатку.

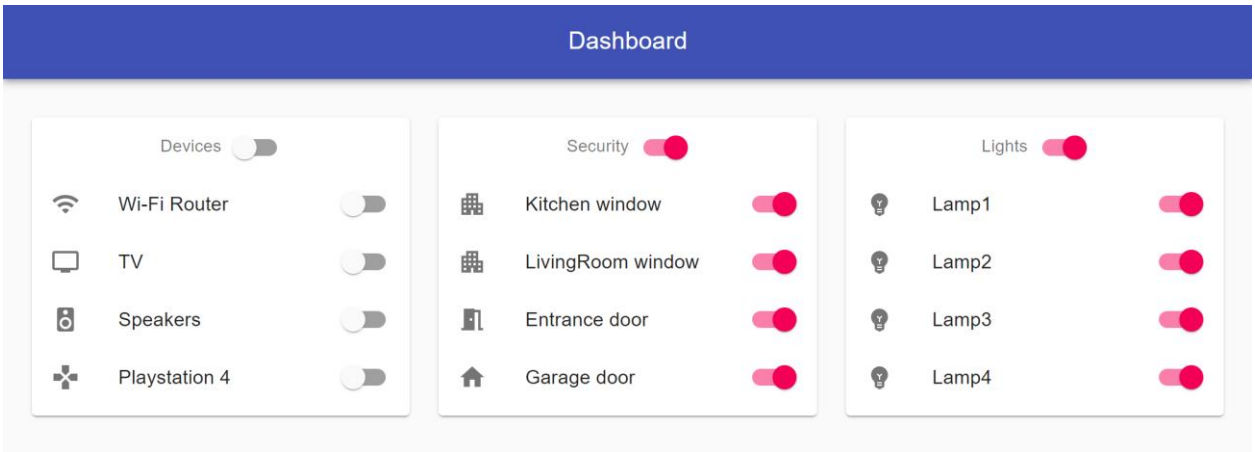


Рисунок 3.3 – Головна сторінка додатку

3.2.4. Реалізація сторінки пристроїв

За допомогою лівого меню, яке зображене на рис. 3.4, користувач може перейти на сторінку усіх пристроїв. На цій сторінці, у нього є можливість переглянути список усіх доданих ним пристроїв, їх ім’я, категорію та статус, увімкнути чи вимкнути той чи інший пристрій. Над списком знаходиться кнопка, яка дозволяє додати новий пристрій. При натисканні на неї, викликається форма, зображена на рис. 3.5, у якій потрібно ввести ім’я та обрати категорію пристрою. На рис. 3.6 зображена сторінка пристроїв.

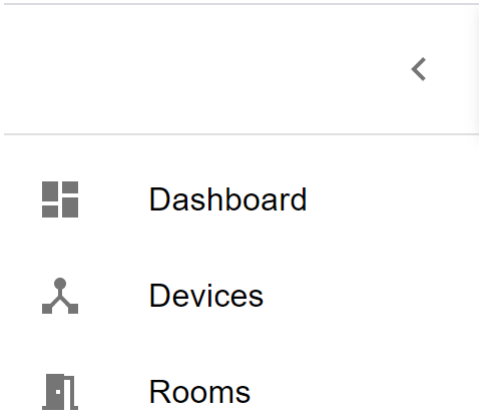


Рисунок 3.4 – Меню додатку

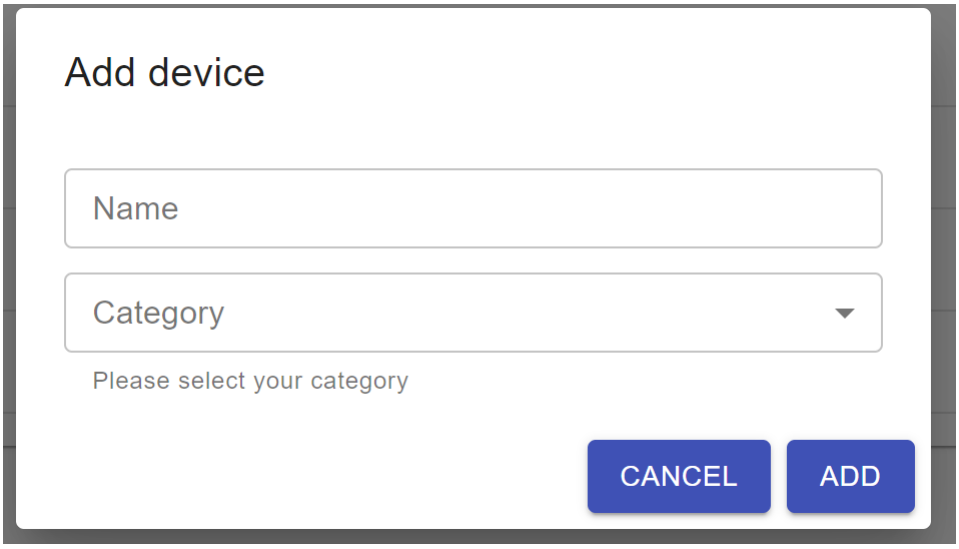


Рисунок 3.5 – Форма додавання пристрою

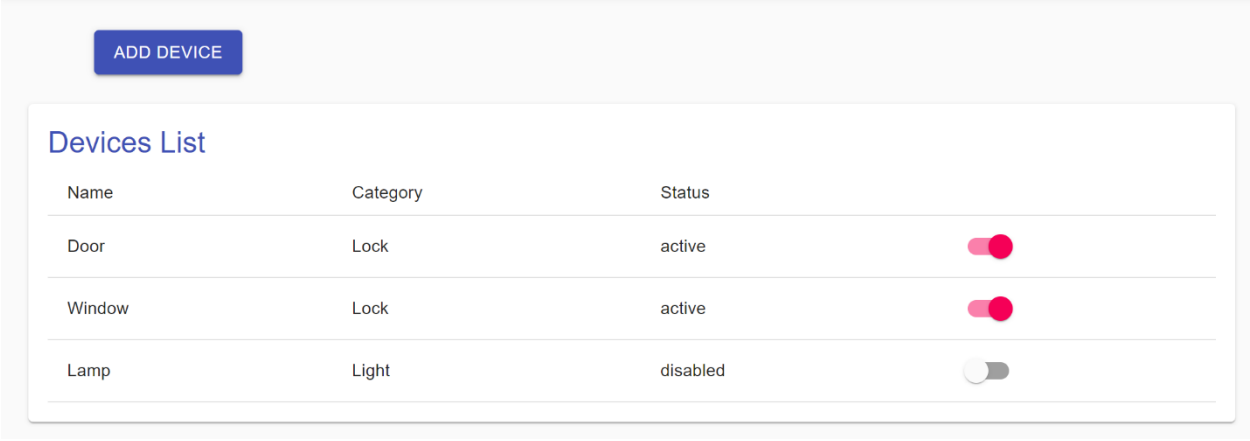


Рисунок 3.6 – Сторінка пристроїв

3.2.5. Реалізація сторінки кімнат

За допомогою лівого меню, яке зображене на рис. 3.4, користувач може перейти на сторінку усіх кімнат. На цій сторінці, у нього є можливість переглянути список усіх доданих ним кімнат, у яких відображаються додані до них пристрої, їх імена, категорії та статус. Над списком знаходиться кнопка, яка дозволяє додати нову кімнату. При натисканні на неї, викликається форма, зображена на рис. 3.7, у якій потрібно ввести назву кімнати. Також є кнопка додавання пристрою до кімнати. На рис. 3.8 зображена сторінка кімнат.

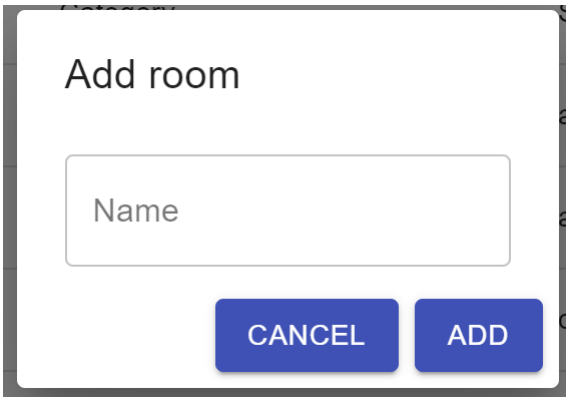


Рисунок 3.7 – Форма додавання кімнати

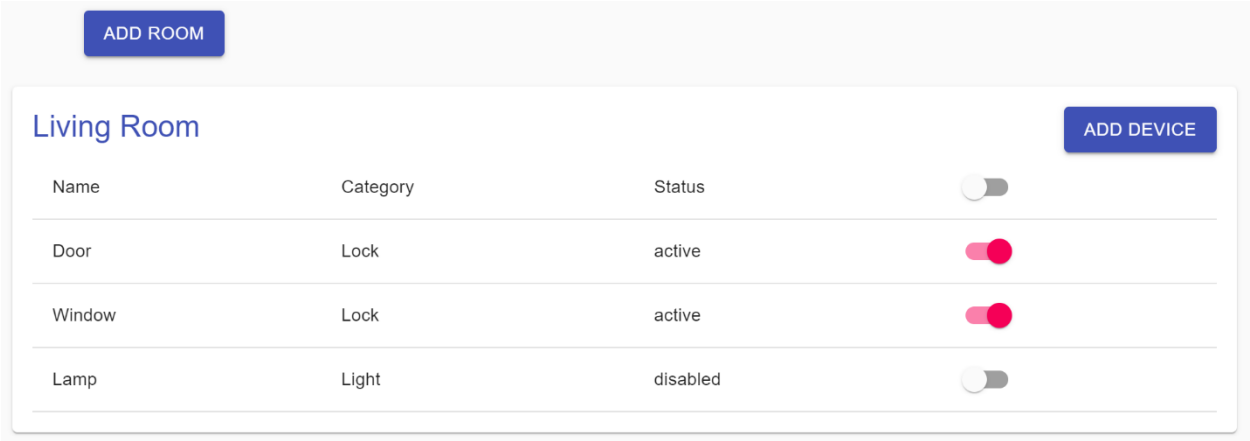


Рисунок 3.8 – Сторінка кімнат

3.3. Структура програми

Веб-додаток системи безпеки розумного дому було створено використовуючи бібліотеку ReactJS. Структурно програма складається з:

головного компоненту App.js, компонентів головної сторінки, сторінки пристроїв, сторінки кімнат та компонентів перемикачів. Структура програми наведена у додатку 1.

Компоненти головної сторінки: Dashboard.js та DashboardContent.js. Dashboard.js описує структуру сторінки за допомогою компонентів бібліотеки Material UI, таких як AppBar(), Toolbar(), List(), IconButton(), MenuButton(). DashboardContent.js відповідає за вміст сторінки, використовуючи компоненти Card(), Grid(), Paper(), Container().

Сторінка пристроїв складається з таких компонентів: AddDeviceButton.js, Devices.js та DevicesPage.js. AddDeviceButton.js описує кнопку, за допомогою якої користувач має можливість додати новий пристрій до системи. Компонент Devices.js описує таблицю, у якій містяться додані користувачем пристрої за допомогою компоненту Table() бібліотеки MaterialUI. DevicesPage.js відповідає за вміст сторінки, використовуючи вже описані компоненти. На рис. 3.9 зображено фрагмент коду DevicesPage.js.

```
<main className={classes.content}>
    <div className={classes.appBarSpacer} />
    <Container maxWidth="lg" className={classes.container}>
        <Grid container spacing={3}>
            <Grid item xs={3}>
                <AddDeviceButton />
            </Grid>
            {/* Devices */}
            <Grid item xs={12}>
                <Paper className={classes.paper}>
                    <Devices />
                </Paper>
            </Grid>
        </Grid>
    </Container>
</main>
```

Рисунок 3.9 – Фрагмент коду DevicesPage.js

Сторінка кімнат містить компоненти AddRoom.js, Rooms.js та RoomsPage.js. AddRoom.js описує кнопку додавання кімнати. Компонент Rooms.js відповідає за відображення таблиці кімнат, у якій містяться створені

користувачем кімнати та додані до них пристрої. RoomsPage.js має аналогічний DevicesPage.js функціонал.

Інші компоненти:

- Card() – поверхня, котра відображає контент і дії, які відносяться до одного об’єкту.
- Switch() – перемикач, який дозволяє вмикати чи вимикати пристрій.
- Button() – кнопка, яка дозволяє додавати пристрій чи кімнату.
- List() – компонент, який дозволяє працювати зі списками.

Натискаючи кнопку додати кімнату чи додати пристрій, у користувача з’явиться форма, зображена на рис. 3.5 чи рис. 3.7 (у залежності яку кнопку користувач натиснув), яку йому потрібно заповнити. За це відповідає компонент бібліотеки Material UI Dialog(). Dialog() – такий тип вікна, яке з’являється перед вмістом програми, щоб повідомити про щось або попросити рішення. На рис. 3.10 зображено фрагмент коду компоненту Dialog().

```
<Button variant="contained" color="primary" onClick={handleClickOpen}>
    Add device </Button>
<Dialog open={open} onClose={handleClose} aria-labelledby="form-dialog-title">
    <DialogTitle id="form-dialog-title">Add device</DialogTitle>
```

Рисунок 3.10 – Фрагмент коду Dialog()

Таблиці для сторінки пристроїв та кімнат створені за допомогою компоненту Table(). Також у таблиці присутній компонент Switch(), який дозволяє вмикати/вимикати пристрої. Фрагмент Table() показано на рис. 3.11.

```
        </TableHead>
        <TableBody>
            {props.rows.map((row) => (
                <TableRow key={row.id}>
                    <TableCell>{row.name}</TableCell>
                    <TableCell>{row.category}</TableCell>
                    <TableCell>{row.status}</TableCell>
                    <TableCell><Switch></Switch></TableCell>
                </TableRow>
            ))}
        </TableBody>
    </Table>
```

Рисунок 3.11 – Фрагмент коду Table()

ВИСНОВОК ДО РОЗДІЛУ 3

У даному розділі був проведений аналіз технологій розробки веб-додатку, а також огляд додаткових бібліотек, які можуть допомогти вирішити поставленні задачі. Враховуючи усі переваги та недоліки кожної технології, було обрано мову програмування та фреймворк, що допоможе реалізувати такий додаток – Javascript і ReactJS.

Реалізацію додатку та його компонентів було розбито на такі основні етапи: реалізація сторінки входу; реалізація головної сторінки додатку; реалізація сторінки всіх пристроїв; реалізація сторінки кімнат. Кожен етап реалізації виконувався згідно вимог, зазначених у технічному завданні. У результаті додаток був побудований для web-версії, але може використовуватися і на мобільних платформах.

Були наведені відповідні рисунки готових сторінок системи безпеки розумного дому для web-версії додатку.

Висновки

Розробка даного дипломного проєкту була присвячена дослідженню можливих варіантів рішень, спрямованих на реалізацію системи безпеки розумного дому з використанням web-socket.

У ході виконання проєкту були розглянуті вже існуючі на даний момент рішення, які реалізують системи розумного дому. На основі цих рішень було складено порівняльну характеристику з урахуванням всіх переваг та недоліків кожної системи.

Також було проаналізовано предметну область даної роботи та визначено основні вимоги і функції додатку, а також наведений список прецедентів та різних сценаріїв функцій, які можуть зустрітися у системі. На основі визначених прецедентів та сценаріїв були побудовані схематичні рисунки та таблиці. Проведено проєктування інтерфейсу та побудовано відповідні макети сторінок додатку.

Зважаючи на вище задані вимоги був проведений аналіз використання існуючих технологій та платформ для реалізації системи. Веб-версія буде реалізована майже у всіх можливих браузерах та відображатися практично однаково. Також було обрано мову написання проєкту, а також проведений опис використаних додаткових бібліотек з обґрунтуванням доцільності їх використання.

Реалізація додатку відбувалася у 5 основних етапів з урахуванням зазначеного у технічному завданні функціоналу та вимог до розробки, а саме реалізацію системи безпеки розумного дому з використанням web-socket.

Список використаної літератури

1. Android [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/docs
2. iOS [Електронний ресурс] – Режим доступу до ресурсу: https://developer.apple.com/documentation/.
3. Google Chrome [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Google_Chrome
4. HTML [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/HTML
5. JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/JavaScript
6. Основи CSS [Електронний ресурс] – Режим доступу до ресурсу: https://html5book.ru/osnovy-css/
7. React documentation [Електронний ресурс] – 2020 – Режим доступу до ресурсу: https://reactjs.org/docs/getting-started.html
8. What are web sockets? [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@dominik.t/what-are-web-sockets-what-about-rest-apis-b9c15fd72aac
9. Розумній дім [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/Розумний_дім
10. React vs Angular vs Vue.js [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/techmagic/reactjs-vs-angular5-vs-vue-js-what-to-choose-in-2018-b91e028fa91d
11. Основи React [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/ruvds/blog/343022/
12. JSX [Електронний ресурс] – Режим доступу до ресурсу: https://webformyself.com/react-js-jsx/

13. JSON Web Token [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/JSON_Web_Token
14. SPA додатки [Електронний ресурс] – Режим доступу до ресурсу: https://wezom.com.ua/blog/chto-takoe-spa-prilozheniya
15. Коротко про ReactJS [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/248799/
16. Web application [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Web_application
17. Internet of Things [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Internet_of_things
18. Virtual DOM [Електронний ресурс] – Режим доступу до ресурсу: https://uk.reactjs.org/docs/faq-internals.html

ДОДАТОК 1

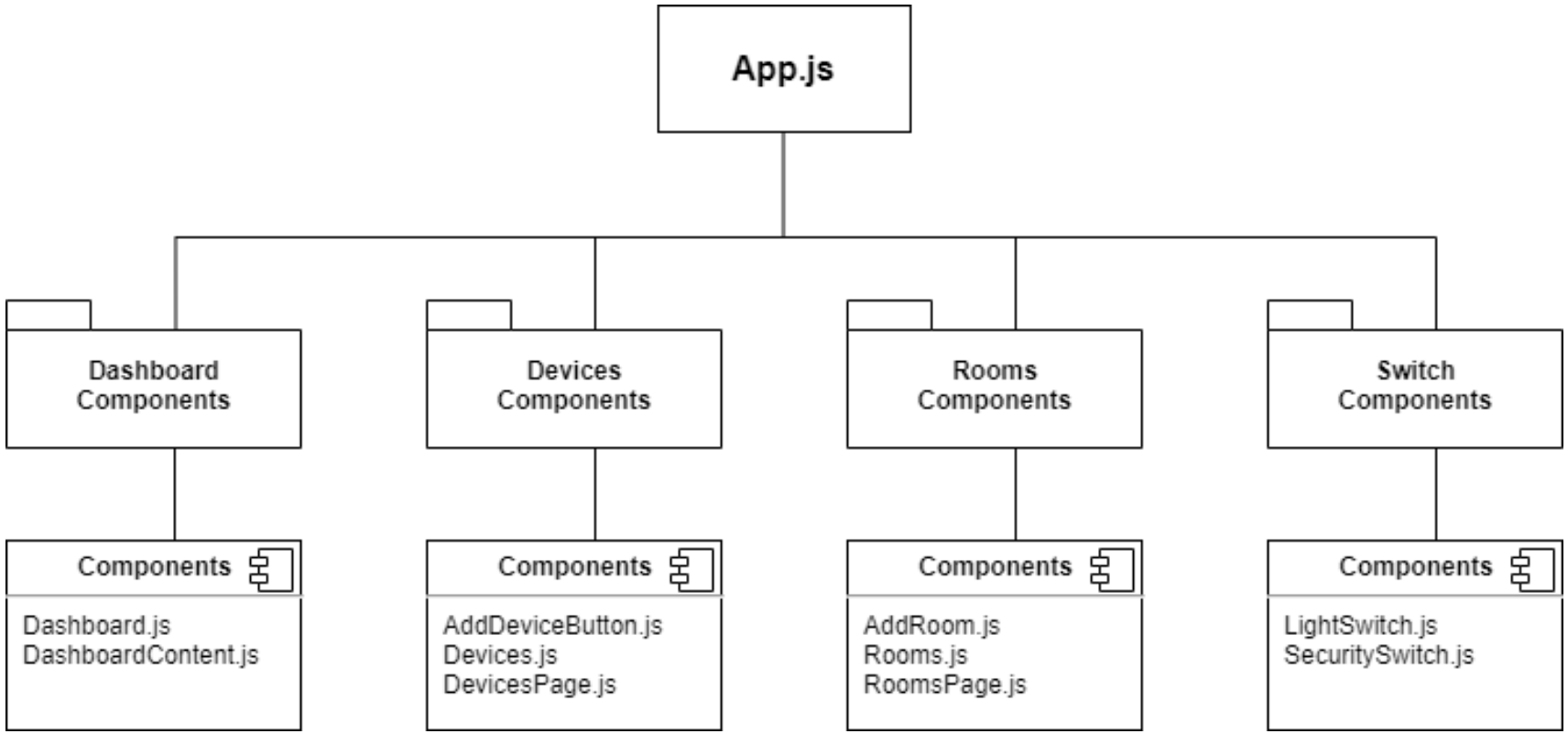
Система безпеки розумного дому з використанням web-socket
(клієнтська частина)

Схема структурна – структура програми

ІАЛЦ.467100.004 Д1

Аркушів 1

Київ – 2020 р.



					ІАЛЦ.467100.004 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Алегрі В.В.			Система безпеки розумного дому з використанням web-socket(клієнтська частина)	Літ. / Аркуш: 1 / Аркушів: 1
Перевірив		Стешин В.В.				
Реценз.						НТУУ «КПІ», ФІОТ, ІО-63
Н. Контр.		Сімоненко В.П.			Схема структурна	
Затв.		Стіренко С.Г.				

ДОДАТОК 2

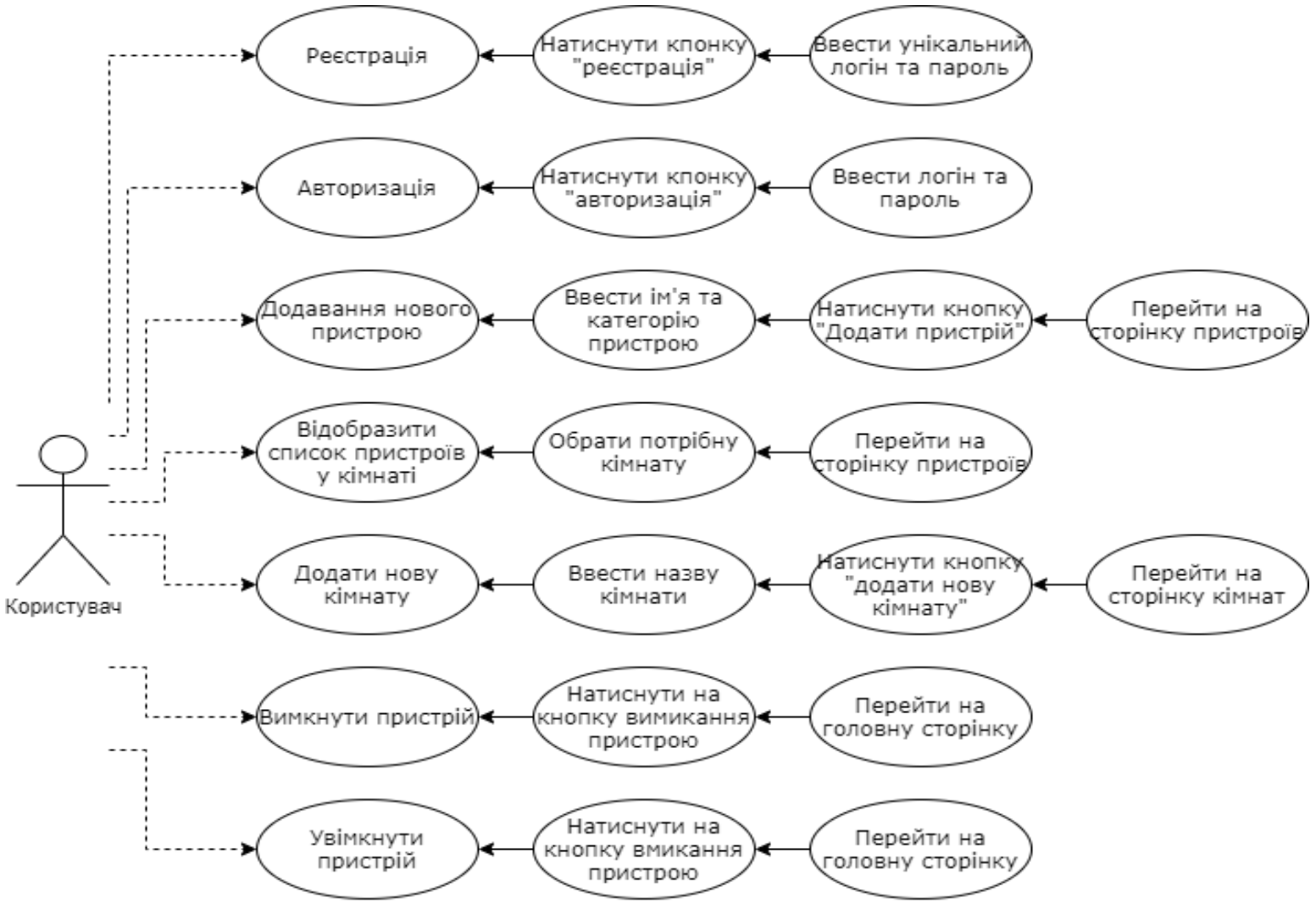
Система безпеки розумного дому з використанням web-socket
(клієнтська частина)

Схема функціональна – схема прецедентів

ІАЛЦ.467100.005 Д2

Аркушів 1

Київ – 2020



					ІАЛЦ.467100.005 Д2	
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розробив		*Алегрі В.В.*			*Система безпеки розумного дому з використанням web-socket(клієнтська частина)*	*Літ.* / *Аркуш* / *Аркушів*
Перевірив		*Стешин В.В.*				*1* / *1*
Реценз.						
Н. Контр.		*Сімоненко В.П.*			*Схема функціональна*	*НТУУ «КПІ», ФІОТ, ІО-63*
Затв.		*Стіренко С.Г.*				

ДОДАТОК З

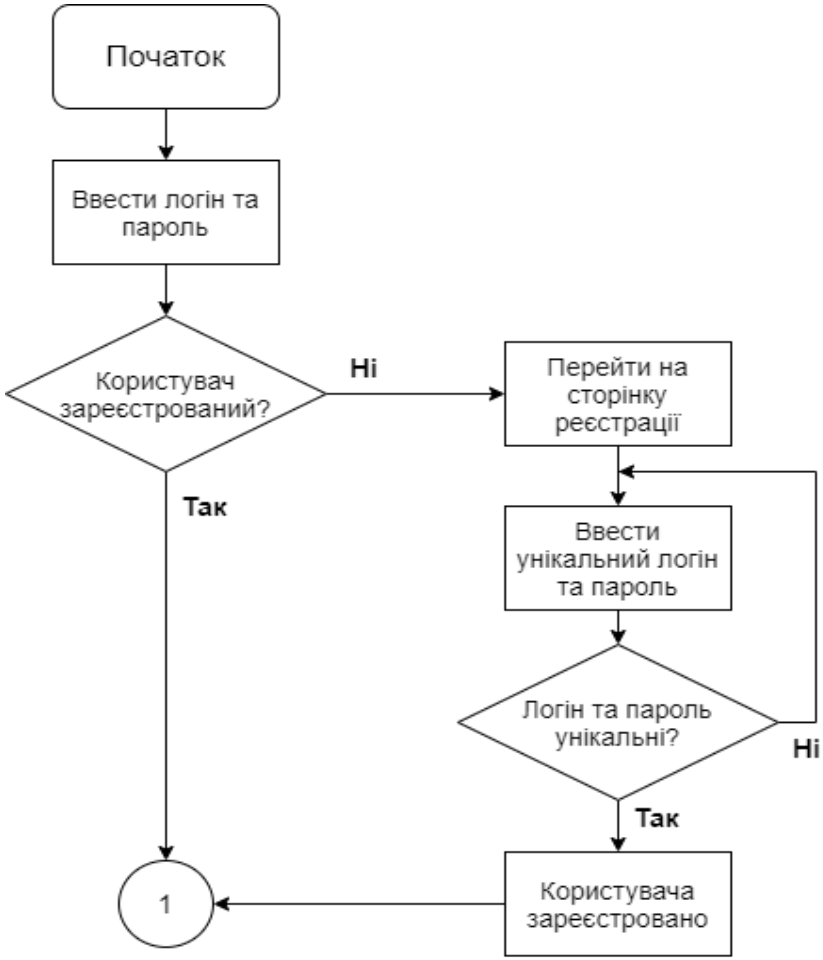
Система безпеки розумного дому з використанням web-socket
(клієнтська частина)

Схема принципова – схема алгоритму додавання нового пристрою

ІАЛЦ.467100.006 ДЗ

Аркушів 1

Київ – 2020



					ІАЛЦ.467100.006 Д3	
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*		
Розробив		*Алегрі В.В.*			*Система безпеки розумного дому з використанням web-socket(клієнтська частина)*	*Літ.* / *Аркуш* / *Аркушів*
Перевірив		*Стешин В.В.*				*1* / *1*
Реценз.						
Н. Контр.		*Сімоненко В.П.*			*Схема принципова*	*НТУУ «КПІ», ФІОТ, ІО-63*
Затв.		*Стіренко С.Г.*				